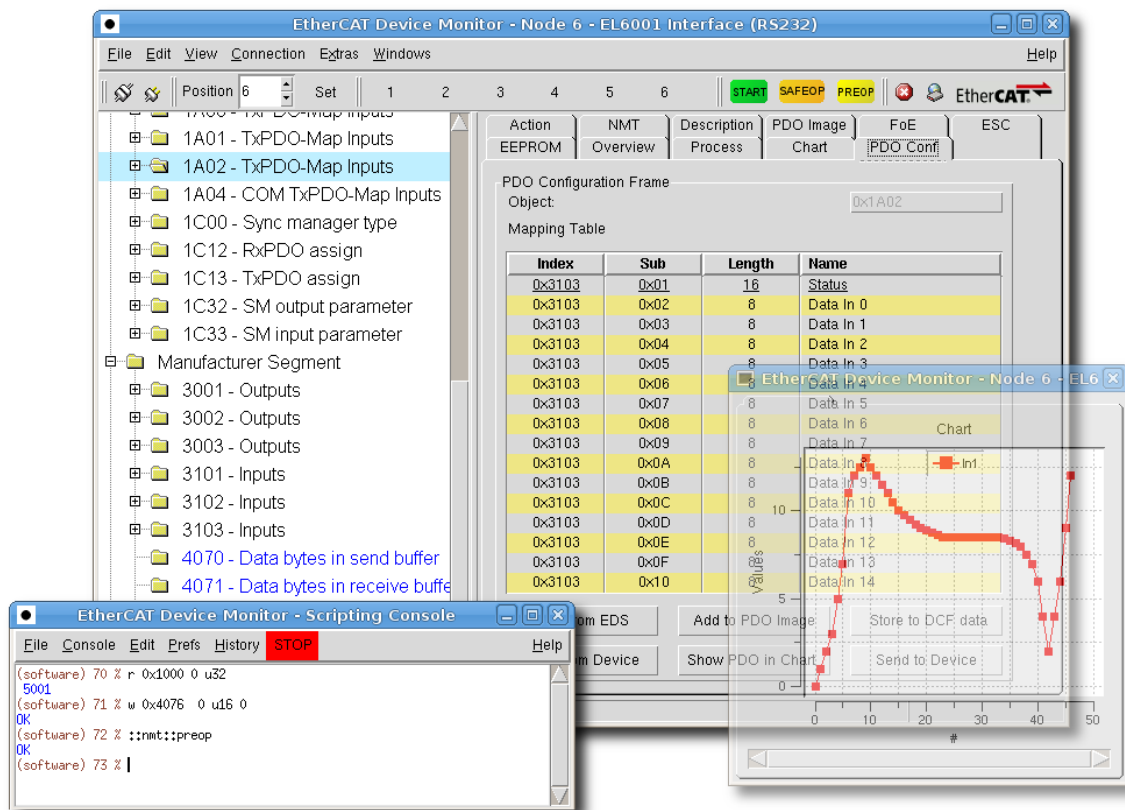


EtherCAT® Device Monitor Benutzerhandbuch



Ablehnungshinweis

Alle Rechte vorbehalten

Die von *port* GmbH gelieferten Programme, Baugruppen und Dokumentationen werden mit großer Sorgfalt erstellt und in unterschiedlichen Einsatzfällen getestet und geprüft.

port GmbH kann trotzdem keine Gewähr oder Haftung dafür übernehmen, dass die Software, die Baugruppe und die Dokumentation fehlerfrei bzw. für spezielle Zwecke geeignet ist.

Insbesondere Beschreibungen und technische Daten sind keine zugesicherten Eigenschaften im rechtlichen Sinne.

Für Folgeschäden, die aufgrund der Benutzung der Programme und Baugruppen auftreten, wird deshalb jede juristische Verantwortung oder Haftung ausgeschlossen.

port hat das Recht, Änderungen an den beschriebenen Produkten oder an der Dokumentation ohne vorherige Ankündigung vorzunehmen, wenn sie aus Gründen der Zuverlässigkeit oder Qualitätssicherung vorgenommen werden oder dem technischen Fortschritt dienen.

Sämtliche Rechte an den Produkten einschließlich der Dokumentation liegen bei *port*. Die Weitergabe an Dritte und Vervielfältigung jeder Art, auch auszugsweise, sind nur mit schriftlicher Genehmigung durch *port* gestattet. Ausgenommen sind Arbeitskopien, die ausschließlich eigenen Zwecken dienen. Dabei trägt der Anwender die Verantwortung, dass die Kopien nicht in den Besitz Dritter gelangen.

Die in dieser Dokumentation verwendeten Soft- und Hardwarebezeichnungen sind in den meisten Fällen auch eingetragene Warenzeichen und unterliegen als solche den gesetzlichen Bestimmungen.

CANopen®

ist eine eingetragene Marke lizenziert durch CiA - CAN in Automation e.V, Deutschland.

EtherCAT®

ist eine eingetragene Marke und patentierte Technologie lizenziert durch die Beckhoff Automation GmbH, Deutschland.

Für Hinweise auf eventuelle Fehler sind wir dankbar und bitten um eine Benachrichtigung.

Wir werden uns bemühen, derartige Hinweise schnellstmöglich zu überprüfen.

Copyright

© 2012 *port* GmbH
Regensburger Straße 7
D-06132 Halle
Tel. +49 345 - 777 55 0
Fax. +49 345 - 777 55 20
E-Mail service@port.de
Internet <http://www.port.de>

Bitdarstellung von numerischen Objekten	40
Nutzerspezifische Registerkarten	41
Registerkarten mit Testunterstützung	41
Registerkarten ohne Inhalt	45
Registerkarten Löschung	46
Konsole	47
Tcl Kommandos	47
Skripts	48
DSP 402 Erweiterung	49
Zustandsmaschinen-Register	49
Profile Velocity Mode Register	50
Profile Position Mode Register	50
Objekterweiterungen	51
About & Release Info Dialog	53
About Dialog	53
Latest Release Info Dialog	53
EDM Command syntax	54
SDO commands	54
Test commands	58
EDM commands	61
DSP402 commands (DSP402-Extension)	68
Glossar	75
Index	76

1.

1.1.

Der EtherCAT® Device Monitor (EDM) kommuniziert mit EtherCAT® Geräten in einem EtherCAT® Netzwerk unter Nutzung der verschiedenen EtherCAT® Dienste. Er unterstützt damit:

- die Entwicklung,
 - die Diagnose,
 - die Inbetriebnahme und Konfiguration
- von EtherCAT® Geräten.

Der EtherCAT® Device Monitor nutzt zur Kommunikation mit den EtherCAT®-Geräten ein EtherCAT®-Server. Die Kommunikation zwischen dem EtherCAT® Device Monitor und EtherCAT® Server vollzieht sich über das TCP/IP Protokoll.

Der EtherCat-Server ist dabei ein EtherCAT-Master und kann sowohl auf dem lokalen PC wie der EtherCAT® Device Monitor laufen als auch auf abgesetzten Geräten.

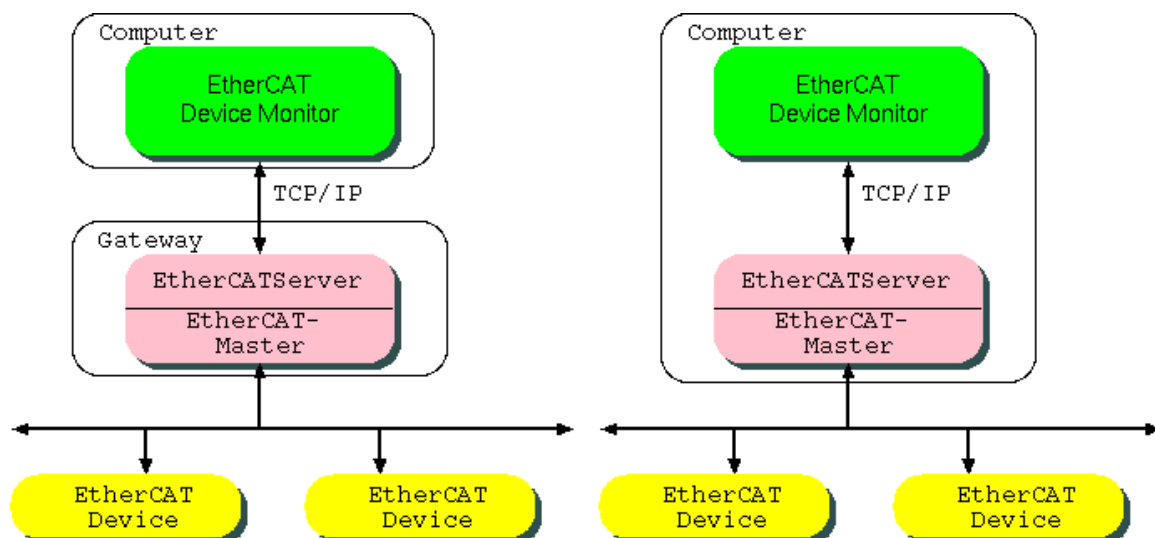


Bild 1: Struktur des EtherCAT® Device Monitors

Der EtherCAT® Device Monitor steht in zwei Ausführungen zur Verfügung:

- als Voll-Version und
- als StarterKit Version.

Die StarterKit-Version besitzt folgende Einschränkungen:

- keine Konsole zur Abarbeitung von Skripts,
- keine Gerätekonfiguration per DCF-Datei,
- Unterstützung nur einiger fester Knotennummern

- keine Unterstützung für FoE und EEPROM-Zugriffe
- begrenzte Laufzeit auf 30 Minuten

Der EtherCAT® Device Monitor besitzt in der Voll-Version die Fähigkeit Skripts mit Kontrollstrukturen zu bearbeiten. Die Erstellung ganzer Master Applikationen ist auf Basis der Sprache [Tcl/Tk](http://www.tcl.tk) (<http://www.tcl.tk>) möglich.

Der Device Monitor ist auf Systemen einsetzbar, die folgende Anforderungen erfüllen:

Betriebssystem:	Windows™, Vista™ UNIX (LINUX)
RAM:	512 MByte
Festplattenspeicher:	25 MByte

1.2.

Im Handbuch werden folgende Konventionen verwendet:

OPERATIONAL	Kommunikationszustände werden groß geschrieben.
tcl_command	Tcl Kommandos werden mit konstant breiten Buchstaben geschrieben. Überschreitet die Länge einer Kommandozeile die Papierbreite, so wird nach Tcl Konvention die Zeile mit einem '\' abgeschlossen und in der Folgezeile fortgesetzt.
beispiel	Beispiele werden mit konstant breiten Buchstaben geschrieben.
0x<wert>	Hexwerte werden durch ein vorgestelltes '0x' gekennzeichnet.
<taste>	Tastaturtasten werden durch < >-Klammern gekennzeichnet.
<set_value>	Im Zusammenhang mit einer Funktionsbeschreibung ist set_value durch einen gewünschten Wert zu ersetzen.
[option]	Im Zusammenhang mit einer Funktionsbeschreibung kann option durch eine Option der Funktion ersetzt werden.
"directory"	Verzeichnisangaben werden durch Anführungszeichen gekennzeichnet.
Console → Save.. → All	Der Weg durch Menüs wird in dieser Form dargestellt.

1.3.

port unterstützt die Anwender mit einer telefonischen Hotline und durch Schulungen. Weiterhin bietet *port* Beratungen im Umfeld von CAN und EtherCAT®, wie z.B. Netzwerkplanung, Netzwerkkonfiguration, Geräteauswahl und EtherCAT® und

CANopen-Geräteprofilimplementierungen an.

2.

2.1.

Der häufigste Anwendungsfall des EtherCAT® Device Monitors ist, dass der EtherCAT-Server local auf dem selben PC wie der EtherCAT® Device Monitor verwendet wird. Vor der ersten Verwendung muss der EtherCAT® Device Monitor so konfiguriert werden, dass der EtherCAT-Server automatisch gestartet wird. Dies erfolgt in unter dem Menüpunkt **Connection -> EtherCAT®-Interface** . In der Auswahlbox "Ethercat-Interface" wird "Local EtherCATserver" eingestellt und darunter muss der Pfad zur Datei `ecatserver.exe` angegeben werden. Abschließend kann in der Auswahlbox "Device" die Ethernet-Schnittstelle ausgewählt werden, an welche das EtherCAT-Netzwerk angeschlossen ist.

Für die zweite Variante, die Verwendung eines EtherCATservers auf einem entfernten Rechner oder embedded-PC, ist TCP als EtherCAT-Interface anzugeben und die IP-Adresse des Rechners bei Host anzugeben.

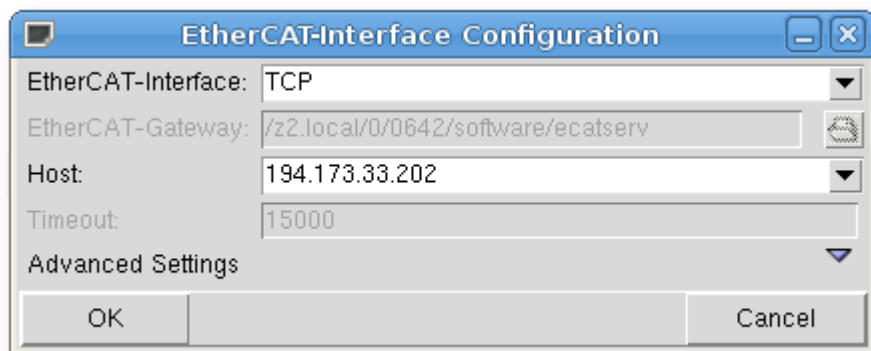


Bild 7, EtherCAT®-Interface-Konfiguration

Ist dies eingetragen, so kann eine Verbindung mit dem EtherCAT®-Gateway hergestellt werden. Klicken Sie dazu auf das erste Symbol in der Werkzeugleiste.

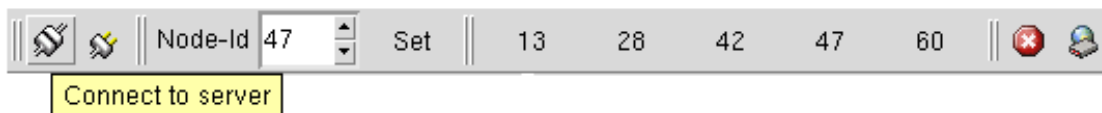


Bild 8, "Connect"-Symbol in der Werkzeugleiste

⚠ Nach der Verbindung prüft das EtherCAT®-Gateway die auf dem Gateway vorhandene Lizenz. Ist die Lizenz ungültig, wird der Anwender über ein Dialogfenster darüber informiert und es kann nur mit einigen wenigen Knoten kommuniziert werden (Demo-Version).

Der EtherCAT® Device Monitor verbindet sich nun mit dem EtherCAT®-Server und bei Erfolg wird der Online-Status in der Statusleiste angezeigt und der Hintergrund des Ein-

gabefeldes färbt sich pink.

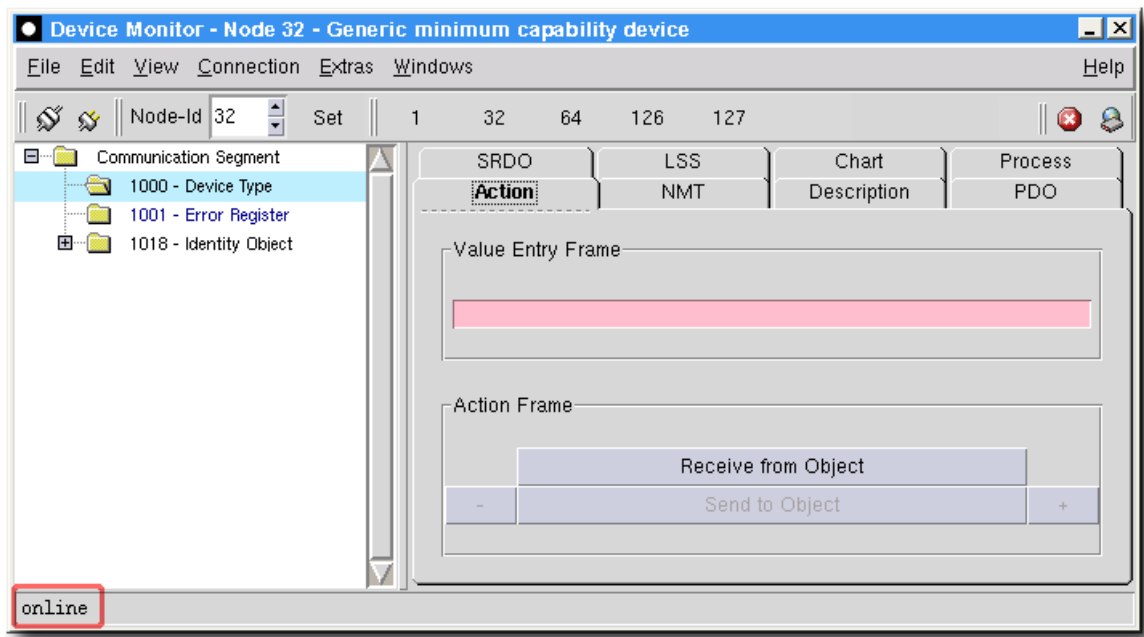


Bild 3: EtherCAT® Device Monitor

! War die Konfiguration des Hardware-Interfaces erfolgreich, speichert der EtherCAT® Device Monitor diese Konfiguration und beim nächsten Start des Programms können diese gespeicherten Einstellung verwendet werden. Zusätzlich kann **Extras → Options → Auto-Connect** aktiviert werden, so dass sich der EtherCAT® Device Monitor bei jedem Neustart des Programms automatisch verbindet.

2.2.

Stellen Sie zuerst die Knotennummer des EtherCAT®-Gerätes in der Werkzeugleiste ein. Bitte beachten Sie, dass die Auswahl mit "Set" bestätigt werden muss.



Bild 4: Toolbar

Als zweiter Schritt kann die EDS-Datei des Gerätes über **File → Load EDS** geladen werden. Alternativ kann mit dem Default-EDS auf die Mandatory-Objekte des Gerätes zugegriffen werden.

Nach der Auswahl eines Objekts im EDS-Baum kann über den "Action-Tab", auf das Objekt, je nach Zugriffstyp, lesend und schreibend zugegriffen werden. Beim Betätigen von "Send to Object" wird nur der **letzte Ausdruck** nach dem letzten Leerzeichen übertragen. Wenn z.B. der EtherCAT® Device Monitor nach dem Senden von "100", "Ok,

wrote 100" in das Eingabefeld schreibt, so reicht es, wenn die "100" verändert wird um einen neuen Wert zu senden. Das "Ok, wrote" braucht nicht extra gelöscht zu werden. Ebenso ist es möglich, mathematische Ausdrücke in das Eingabefeld zu schreiben. Der EtherCAT® Device Monitor berechnet diese und überträgt das Ergebnis zum Gerät. Der Ausdruck darf jedoch keine Leerzeichen enthalten.

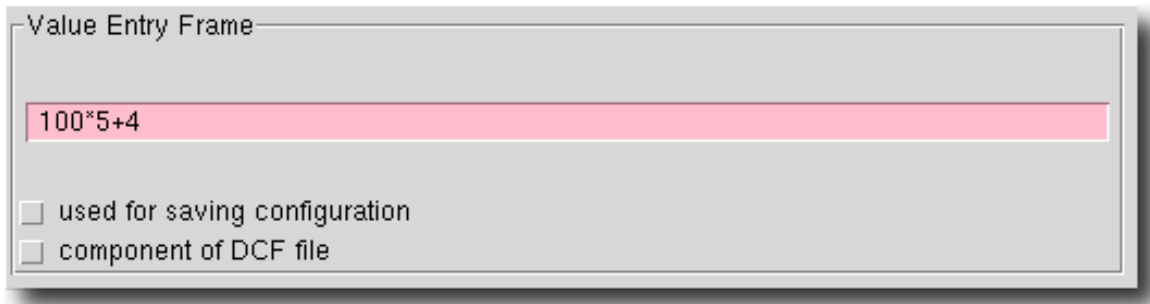


Bild 5: Formeln im Eingabefeld

Zum Senden von NMT-Kommandos dient der "NMT-Tab". Dieser kann, wie jeder Tab, aus der Verankerung im EtherCAT® Device Monitor heraus in ein separates Fenster gezogen werden. Klicken Sie dazu auf die gestrichelte Linie unterhalb des Reiters.

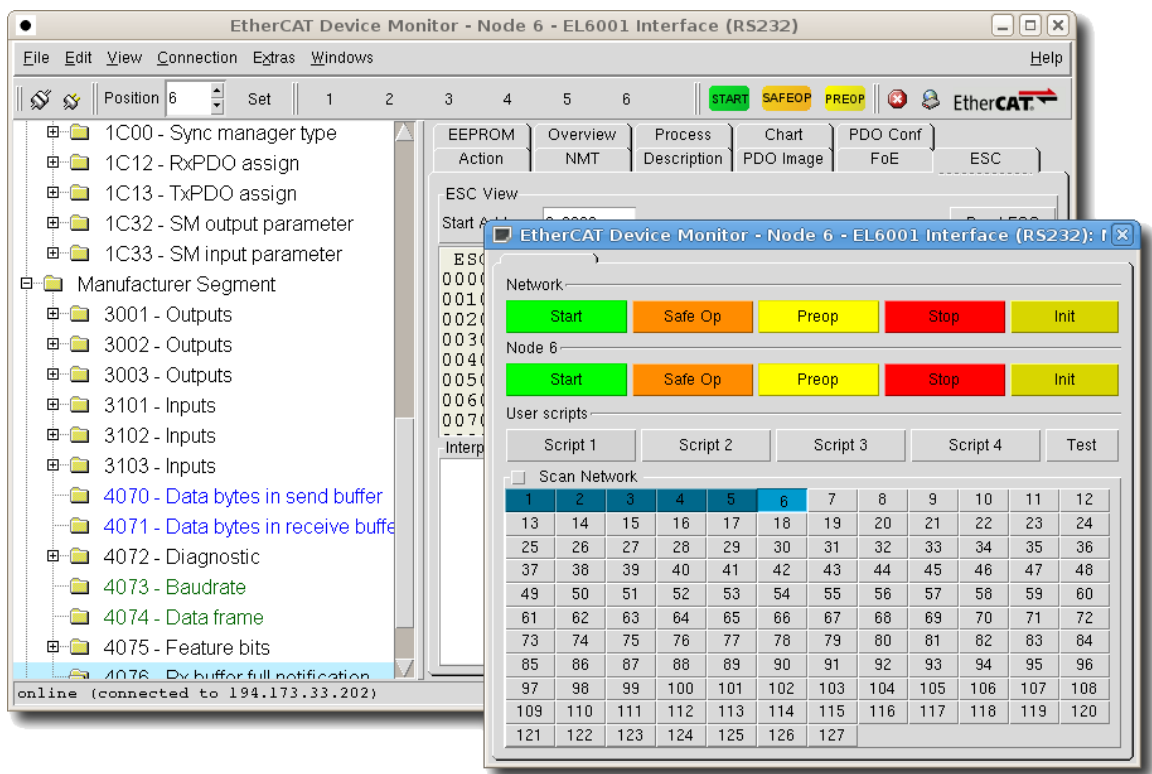


Bild 6: Herauslösen eines Tabs

Die obere Kommandoleiste dient zum Senden der NMT-Kommandos an das ganze Netzwerk und über die untere wird jeweils der aktuelle Knoten angesprochen. Der aktuelle Knoten kann über das Auswahlfeld im unteren Teil des NMT-Tabs schnell geändert werden.

3.

Das Objektverzeichnis ist die Datenschnittstelle zwischen dem EtherCAT® Device Monitor und einem EtherCAT® Gerät. Auf jedes Objekt im Objektverzeichnis kann der EtherCAT® Device Monitor über einen Index/Subindex zugreifen.

Das aus dem Electronic Data Sheet (ESI-Datei) geladene Objektverzeichnis wird als Baumstruktur im EtherCAT® Device Monitor angezeigt. Alle Objekte, die als ARRAY oder RECORD definiert sind und mehrere Subindizes beinhalten, werden durch ein Kreuz markiert.

Für jedes Gerät im Netzwerk kann ein eigenes ESI geladen werden. Nach der Auswahl eines Gerätes wird das Objektverzeichnis des aktiven Gerätes dargestellt und es kann mit dieses Gerät kommuniziert werden.

Mittels der nachfolgend dargestellten Registerkarte **Action** kann auf jedes Objekt lesend oder schreibend per SDO zugegriffen werden. Dafür werden die SDO-Protokolle *Read by Index* und *Write by Index* verwendet.

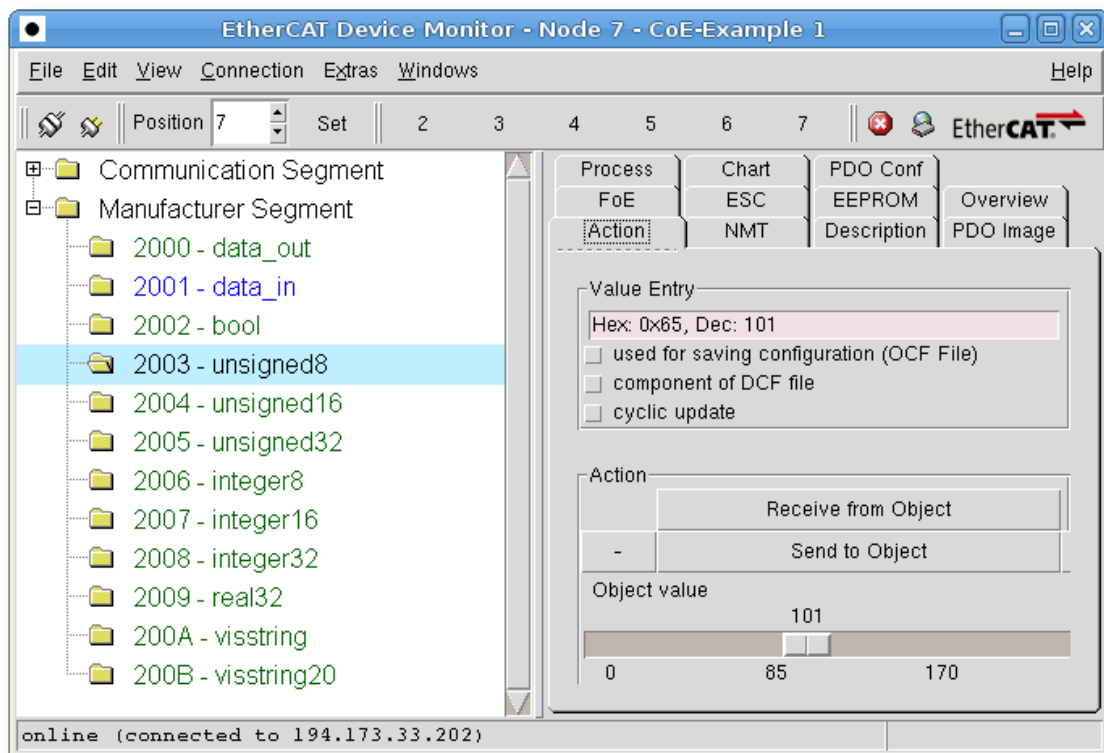


Bild 15: Registerkarte Action

Objekt setzen

Objekt in der Baumstruktur auswählen

Wert in Action-Tab → Value Entry Frame eintragen

SDO Transfer auslösen durch Action → Send to Object

Objekt lesen

Objekt in der Baumstruktur auswählen

SDO Transfer auslösen durch Actions → Receive from Object

der empfangene Wert wird in Actions → Value Entry Frame angezeigt

Die Auswahl eines Objekts im Objektverzeichnis löst noch keinen Lesevorgang aus. Ist dies jedoch explizit gewünscht, so ist Extras → Options → Read object on selection zu aktivieren.

Value Entry Frame

Hexadezimale Werteangaben werden mit einem 0x gekennzeichnet (z.B.: 0x10).

Strings, auch wenn sie aus mehreren Teilen bestehen, werden ohne Anführungszeichen eingegeben. Abschließende Leerzeichen werden verworfen.

Ist dieses Feld leer, wird der Wert 0 gesendet.

<DEL> löscht den Inhalt dieses Feldes.

<ENTER> sendet den Wert.

<Shift ENTER> befindet sich im Feld eine URL, wird der Webbrowser gestartet.

Das letzte Wort im Eingabefeld wird als mathematischer Ausdruck ausgewertet, bevor es dem SDO Transfer übergeben wird.

Probieren Sie folgendes aus: $0x180+10$

Component of DCF file

Über dieses Feld wird das Objekt mit der DCF-Datei verknüpft (siehe Kapitel "Datenverwaltung per DCF-Datei").

Used for saving configuration

Über dieses Feld kann das Objekt für die Speicherung in einer Objektkonfigurations-Datei (*.ocf) ausgewählt werden (siehe Kapitel "Datenverwaltung per OCF Datei").

Cyclic update

Ist "cyclic update" aktiviert, so wird der Wert des Objekts zyklisch gelesen und aktualisiert. Aktive Bitboxen und Slider, welchen den Wert des Objekts darstellen, werden in diesem Fall auch aktualisiert.

Receive from Object

Der Wert eines Objektes wird per SDO abgefragt.

Send to Object

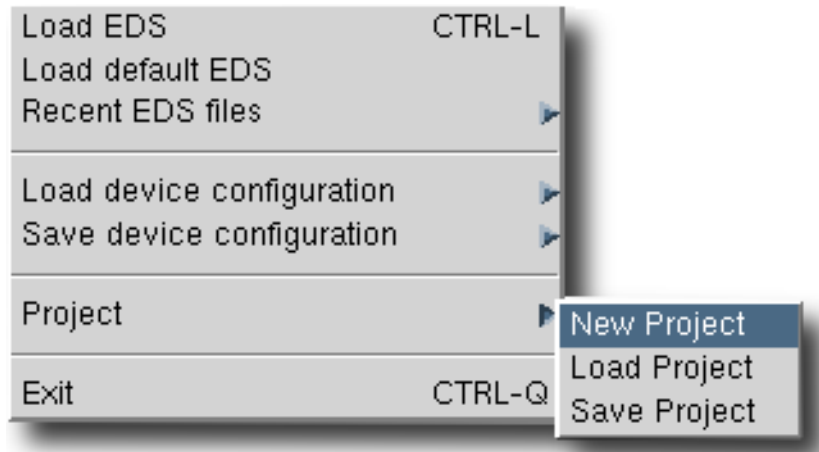
Der Wert eines Objektes wird per SDO gesetzt.

- Der Wert im Feld Value Entry Frame wird um den Wert 1 verringert und automatisch an das Gerät gesendet.

- + Der Wert im Feld Value Entry Frame wird um den Wert 1 erhöht und automatisch an das Gerät gesendet.

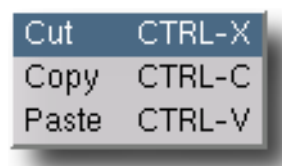
4.

4.1.



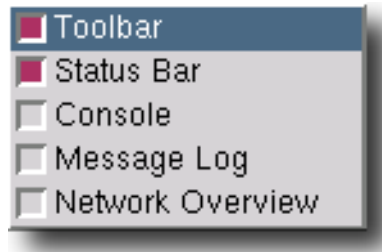
<i>Load EDS</i>	Lädt eine EDS-Datei aus dem Repository oder einem beliebigen Verzeichnis
<i>Load default EDS</i>	Lädt eine EDS-Datei mit Mandatory-Objekten
<i>Recent EDS files</i>	Liste kürzlich verwendeter EDS-Dateien
<i>Load device configuration</i>	Lädt OCF oder DCF-Dateien
<i>Save device configuration</i>	Speichert OCF oder DCF-Dateien
<i>New Project</i>	Startet ein neues Projekt
<i>Load Project</i>	Lädt ein vorhandenes Projekt
<i>Save Project</i>	Speichert ein Projekt
<i>Exit</i>	Beendet das Programm

4.2.



<i>Cut</i>	Schneidet eine Auswahl aus einem Eingabe- oder Textfeld aus
<i>Copy</i>	Kopiert eine Auswahl aus einem Eingabe- oder Textfeld
<i>Paste</i>	Fügt eine Auswahl in ein Eingabe- oder Textfeld ein

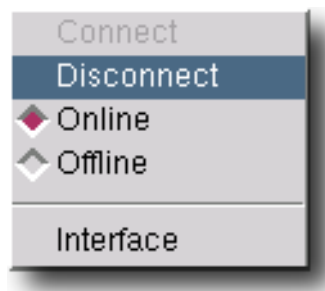
4.3.



Toolbar
Status Bar
Console

Schaltet die Ansicht der Werkzeugleiste ein oder aus
 Schaltet die Ansicht der Statusleiste ein oder aus
 Schaltet die Ansicht der Konsole ein oder aus (nur Vollversion)

4.4.



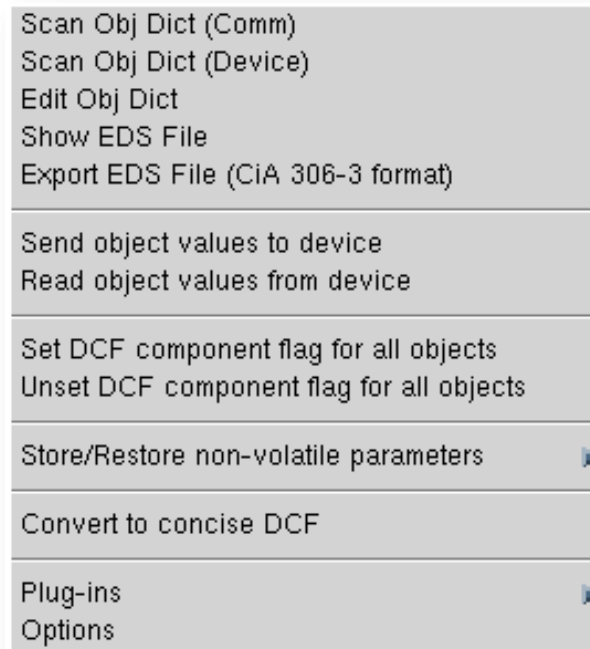
Connect
Disconnect
Online

Offline

Interface

Stellt eine Verbindung zum EtherCAT®-Server her
 Trennt die Verbindung zum EtherCAT®-Server
 Schreib- und Lesezugriffe kommunizieren direkt mit dem Gerät
 Schreib- und Lesezugriffe erfolgen mit dem DCF Datenpuffer
 Öffnet das Dialogfenster zur Konfiguration des EtherCAT®-Interface

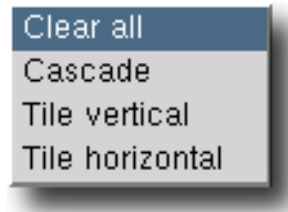
4.5.



<i>Scan Obj Dict (Comm)</i>	Scannt den Kommunikationsprofilbereich des Objektverzeichnisses
<i>Scan Obj Dict (Device)</i>	Scannt den Geräteprofilbereich des Objektverzeichnisses
<i>Edit Obj Dict</i>	Öffnet einen einfachen OD-Editor. Dieser ist beispielsweise zum Einfügen von dynamischen OD-Einträgen sehr sinnvoll.
<i>Export EDS File</i>	Exportiert ein gescanntes Objektverzeichnis als EDS-Datei. ¹
<i>Send object values to device</i>	Überträgt die Werte aller Objekte mit DCF-Component-Flag zum Gerät.
<i>Read object values from device</i>	Liest die Werte aller Objekte mit DCF-Component-Flag vom Gerät.
<i>Store/Restore non-volatile parameters</i>	Speichert oder lädt die Gerätekonfiguration im nicht-flüchtigen Speicher des Geräts
<i>Convert to concise DCF</i>	Konvertiert DCF-Dateien in das Concise-Format
<i>Plug-ins</i>	Menü zum Laden von Plug-ins
<i>Options</i>	Öffnet den Optionen-Dialog

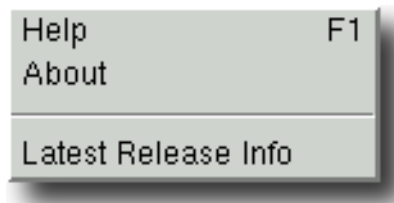
¹ Die exportierte Datei kann vom EtherCAT® Device Monitor wieder eingelesen werden. Jedoch ist die Datei keine vollständige EDS-Datei nach dem Standard.

4.6.



<i>Clear All</i>	Löscht den Inhalt der Console und des Log-Fensters	} nicht das Hauptfenster
<i>Cascade</i>	Ordnet alle offenen Fenster hintereinander an	
<i>Tile vertical</i>	Ordnet alle offenen Fenster untereinander an	
<i>Tile horizontal</i>	Ordnet alle offenen Fenster nebeneinander an	

4.7.



<i>Help</i>	Zeigt die Online-Hilfe an
<i>About</i>	Zeigt Versions- und Lizenzinformationen an
<i>Latest Release Info</i>	Holt Informationen über Updates aus dem Internet

5.

Die Werkzeugleiste wird in der nachfolgenden Abbildung erläutert.



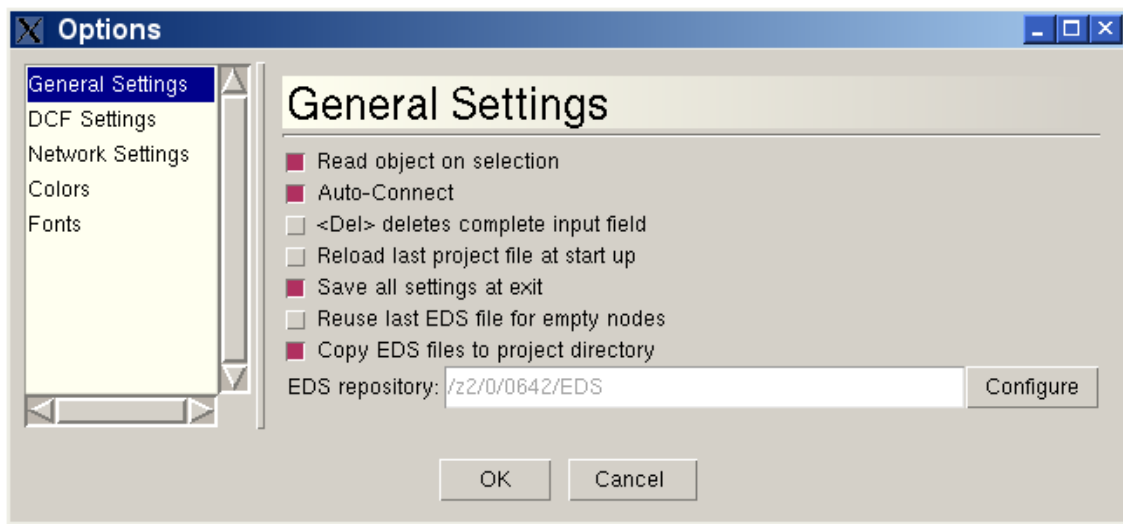
Bild 16: Werkzeugleiste

Die nachfolgende Tabelle beschreibt die Funktionen der Werkzeugleiste von links nach rechts:

Icon	Bedeutung
<i>Connect</i>	Verbindung mit EtherCAT-Netzwerk (EtherCAT-Server) herstellen
<i>Disconnect</i>	Verbindung trennen
<i>Position</i>	Auswahl des aktiven Knotens anhand der Position
<i>1 ... 2 ...</i>	Schnellwahl der letzten aktiven Knoten
<i>Start</i>	Zustandswechsel nach OPERATIONAL für alle Slaves
<i>SafeOp</i>	Zustandswechsel nach SAFE-OPERATIONAL für alle Slaves
<i>PreOp</i>	Zustandswechsel nach PRE-OPERATIONAL für alle Slaves
<i>Refresh</i>	Reinitialisiert den EtherCATserver und aktualisiert die PDO-Konfiguration aller Slaves. Nach einer Änderung des PDO-Mappings oder PDO-Assignments ist dies erforderlich. Dies kann zudem nötig während der Entwicklung eines EtherCAT-Geräts nötig sein, wenn die CPU im Debugger angehalten wird und der ESC jedoch weiterläuft. Der EtherCATserver erkennt in diesem Fall den Neustart des Slaves nicht und muss reinitialisiert werden.
<i>Abort</i>	Beendet länger dauerte Prozesse, siehe auch <code>global_stop</code>
<i>Query</i>	Holt Informationen über Updates aus dem Internet
<i>EtherCAT</i>	Öffnet the EtherCAT-Seite auf der port-Webseite

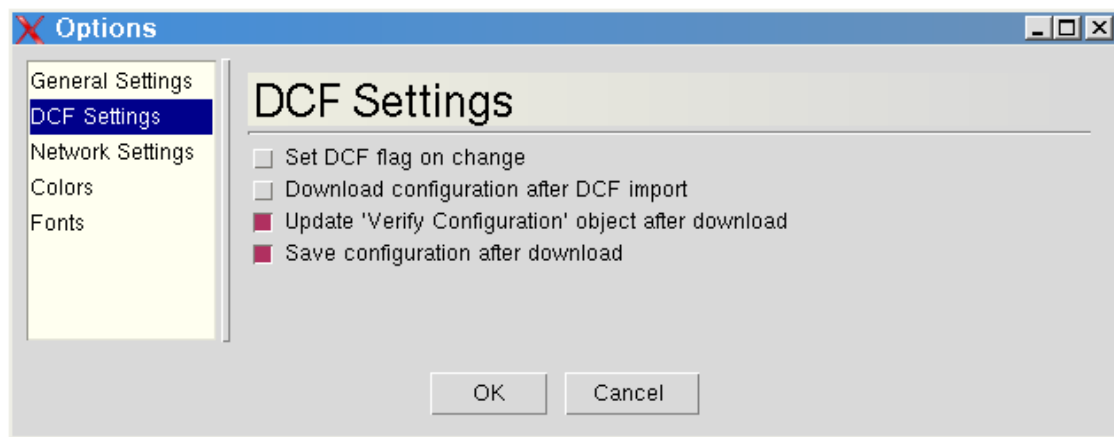
6.

6.1.



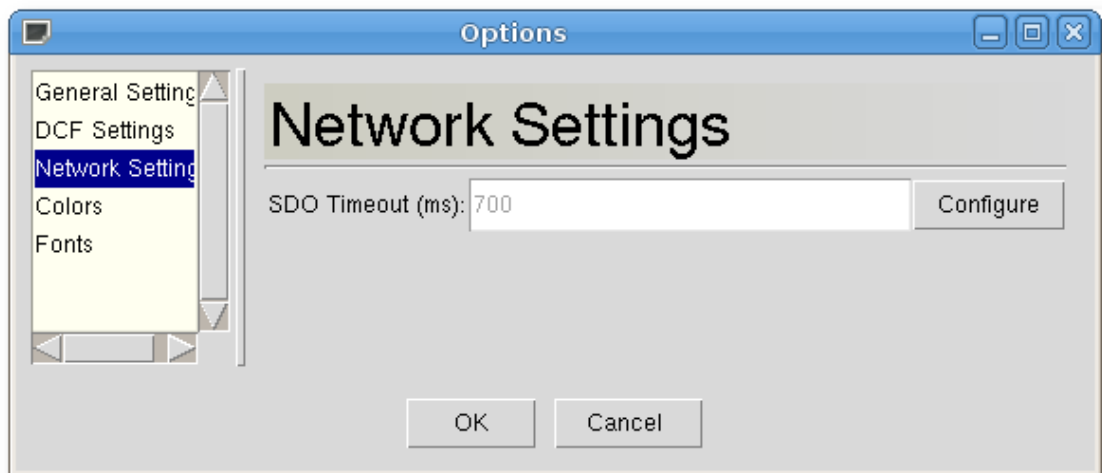
Read object on selection	<i>automatisches Lesen des Objektes bei Auswahl</i>
Auto-Connect	<i>Automatisches Verbinden beim Start des Programms</i>
<Del> deletes complete entry field	<i>Ist dies ausgewählt, wird beim Betätigen der <Del>- bzw. <Entf.>-Taste die gesamte Eingabezeile gelöscht.</i>
Reload last project file at start up.	<i>Automatisches Öffnen der zuletzt verwendeten Projektdatei beim Start des CDM.</i>
Save all settings at exit	<i>Automatisches Speichern aller Einstellungen (Optionen und Connection) beim Beenden des Programms.</i>
Reuse last EDS for empty nodes	<i>Übernahme der letzten EDS-Datei bei Knotenwechsel zur Knoten ohne EDS-Datei.</i>
Copy EDS files to project directory	<i>Kopieren der EDS-Dateien vom Repository in das Projektverzeichnis, wenn ein Projekt gespeichert wird.</i>
EDS repository	<i>Konfiguration des Verzeichnisses für das EDS-Repository, die EDS-Datei-Datenbank des EtherCAT® Device Monitors</i>

6.2.



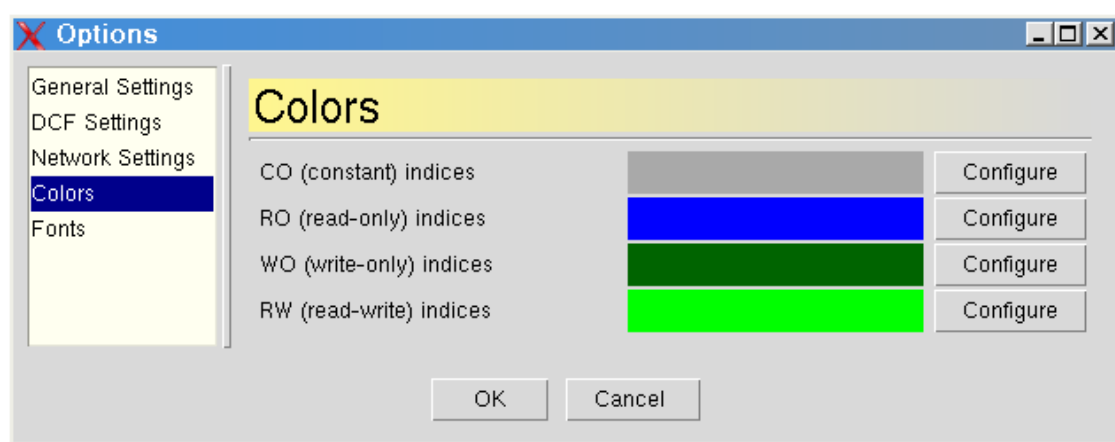
Set DCF flag on change	<i>Setzt das DCF-Component-Flag bei Änderung eines Objekts</i>
Download configuration after DCF import	<i>Automatisches Herunterladen der Konfiguration nach einem DCF-Import</i>
Save configuration after download	<i>Automatisches Speichern der Konfiguration nach dem Herunterladen der Konfiguration</i>

6.3.



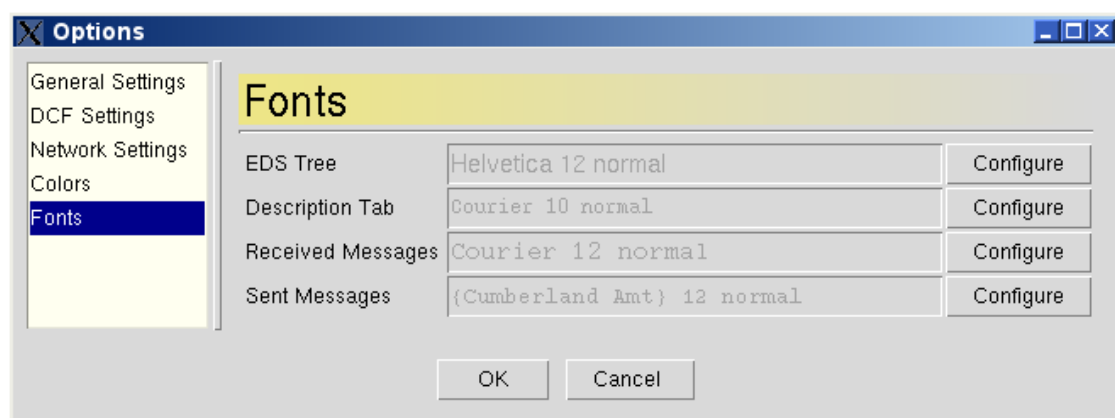
SDO Timeout (ms)	<i>SDO Timeout in ms</i>
EtherCAT Cycle Time(ms)	<i>Zykluszeit des EtherCAT-Netzwerks in ms</i>
Allow old-style OD Scan	<i>Falls das Scannen des Objektverzeichnisses durch SDO-Info nicht unterstützt wird, so kann auch versucht werden, jedes Objekt einzeln zu lesen.</i>
Auto re-scan of network	<i>Automatisches Neuscannen des Netzwerks nach Änderung der Topology</i>

6.4.



Mit diesem Dialog sind die Farben für verschiedene Objekte mit unterschiedlichen Zugriffsattributen einstellbar.

6.5.



Mit diesem Dialog sind die Fonts für verschiedene GUI-Elemente einstellbar.

7.

7.1.

Für die Ausführung von NMT Diensten bietet diese Registerkarte folgende Bedienfelder an:

Schaltfläche	Bedeutung
Start Network	Alle Netzwerkteilnehmer in den Zustand OPERATIONAL schalten.
Start Node	Den ausgewählten EtherCAT®-Knoten in den Zustand OPERATIONAL schalten.
Preop Network	Alle Netzwerkteilnehmer in den Zustand PRE-OPERATIONAL schalten.
Preop Node	Den ausgewählten EtherCAT®-Knoten in den Zustand PRE-OPERATIONAL schalten.
Stop Network	Alle Netzwerkteilnehmer in den Zustand STOPPED schalten.
Stop Node	Den ausgewählten EtherCAT®-Knoten in den Zustand STOPPED schalten.
Safe Op	Setzt den Knoten in den Zustand SAFE OP.
Init	Das NMT-Kommando Init Node senden.

Eine Bedienung der Zustandsmaschine über die Konsole ist auch möglich.

7.2.

Über die Schaltflächen "Script 1" bis "Script 4" können jeweils Skripte mit den Namen script1.tcl bis script4.tcl aus dem Arbeitsverzeichnis oder aus dem Programmverzeichnis gestartet werden. Über "Test" wird das Skript t_start.tcl gestartet, falls die Datei im Arbeitsverzeichnis vorhanden ist. Diese 5 Skripte können frei modifiziert und spezifischen Anforderungen angepasst werden. Bleibt der Mauszeiger eine Sekunden auf der jeweiligen Schaltfläche, so wird als Tool-Tip die erste Zeile der entsprechenden scriptx.tcl Datei angezeigt. Somit erleichtert ein Kommentar zum Dateinhalt in der ersten Zeile des Scripts das Auffinden der gewünschten Datei. Zudem wird das letzte Wort in der ersten Zeile als Farbangabe für den Button verwendet. Innerhalb der Skripte können alle EDM-Kommandos verwendet werden.

Diese Funktion ist nur in der Vollversion verfügbar. In der Eval-Version werden integrierte Skripte geladen.

7.3.

Farbe	Bedeutung
Hintergrundfarbe	Kein Knoten gefunden
blau	Knoten gefunden
gelb	aktiver Knoten während des Netzwerk-Scans

Über die Schaltflächen für die Knoten kann der aktive Knoten auf einfache Weise ausgewählt werden. Wurde für diesen Knoten bereits eine EDS-Datei geladen, so wird diese nun im EDS-Baum dargestellt.

Allen Knoten können jeweils spezifische EDS-Dateien zugeordnet werden.

7.4.

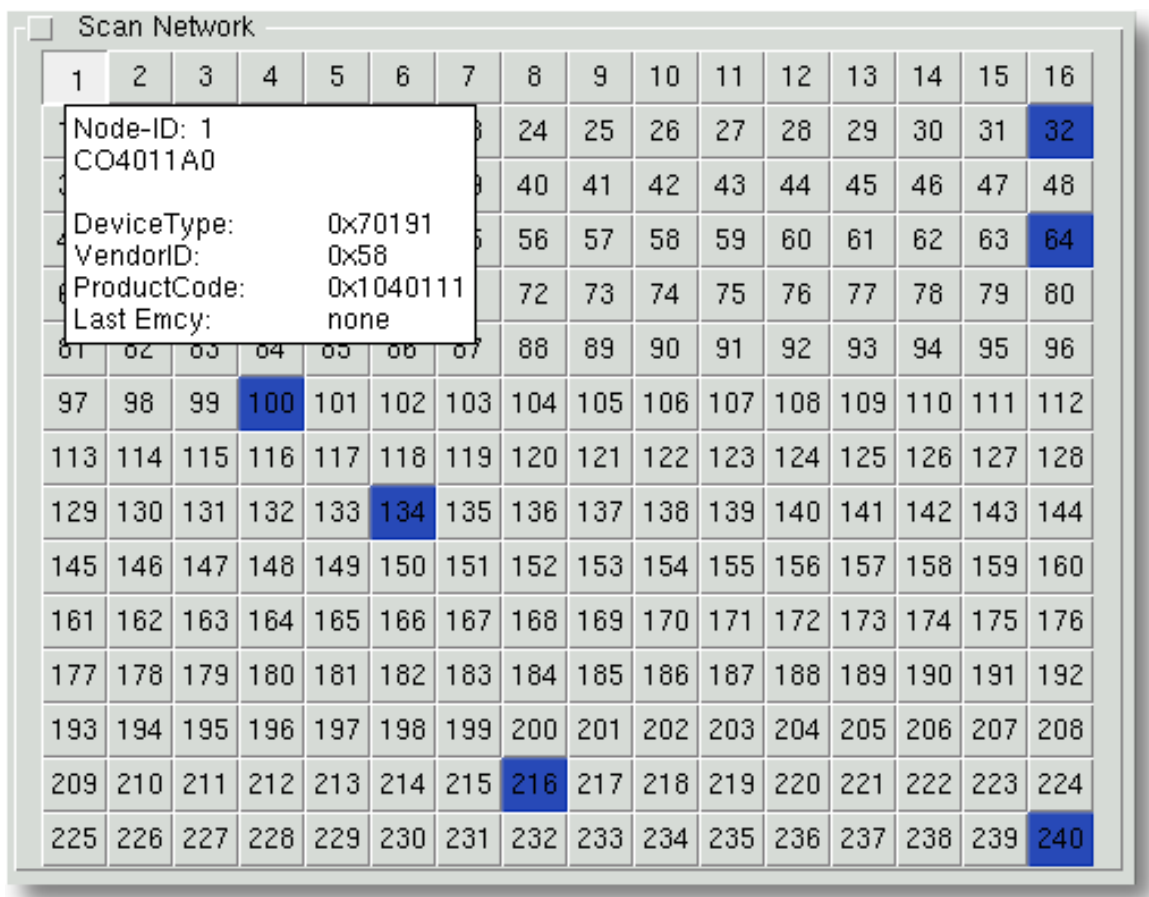


Bild 17: Geräteinformation

Beim Bewegen der Maus über gefundene Knoten im Netzwerk werden Geräteinformation als Tooltip angezeigt.

8.

8.1.

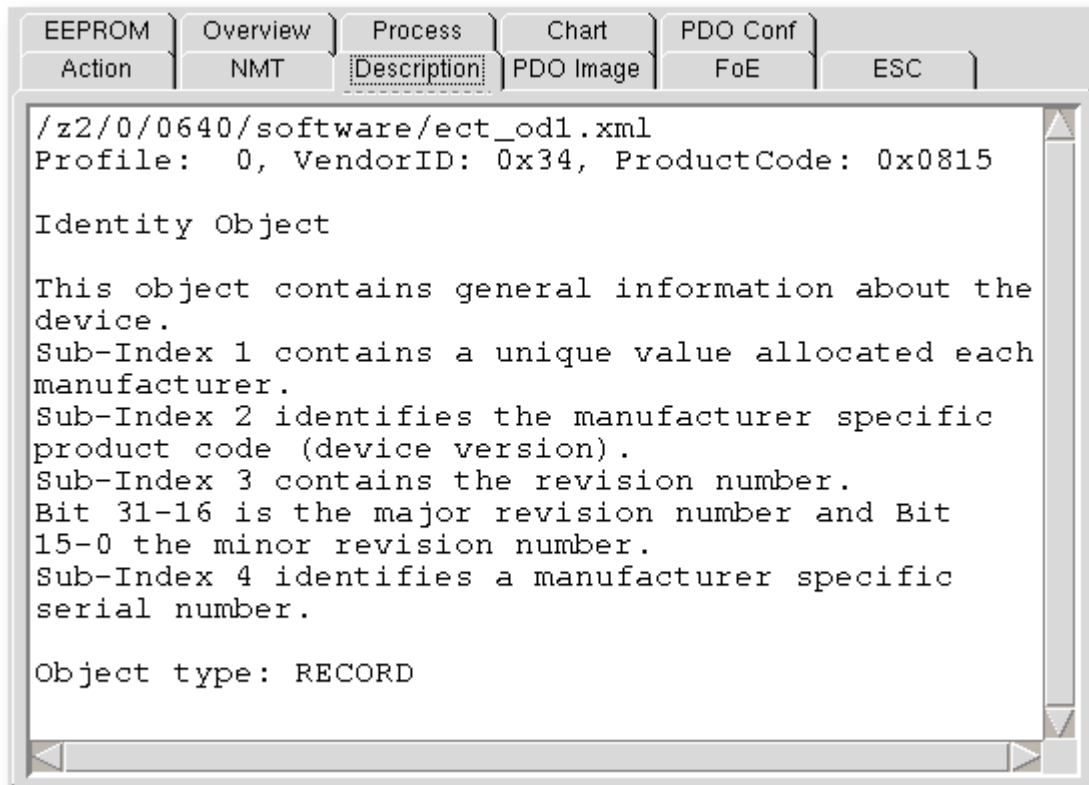


Bild 18: Objektbeschreibung

Die Objektbeschreibung enthält zusätzliche Informationen zu dem ausgewählten Index des Objektverzeichnisses. Neben der eigentlichen Beschreibung wird zusätzlich der Objektcode bei komplexen Datentypen oder der Datentyp und der Defaultwert bei einfachen Datentypen angezeigt. Die Objektbeschreibungen für die einzelnen Indizes des Objektverzeichnisses werden aus den Objektbeschreibungsdateien geladen.

8.2.

Da das alte, auf CANopen basierende EDS-Datenformat keine erweiterten Beschreibungen zu den einzelnen Objekten zulässt, werden die Beschreibungen in einer separaten Datei verwaltet. Die Struktur einer Beschreibungsdatei ist wie folgt:

index1:

Objektname 1

Beschreibungstext 1. Zeile

Beschreibungstext 2. Zeile

Beschreibungstext n. Zeile

index2:
Objektname2

Beschreibungstext 1. Zeile
Beschreibungstext 2. Zeile
Beschreibungstext n. Zeile

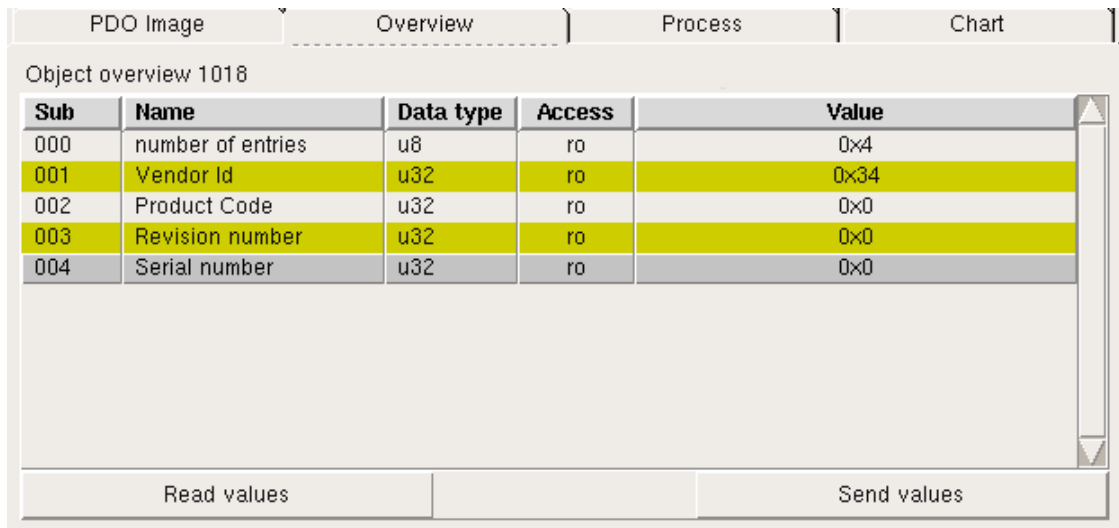
Die Datenformat der Indizes ist hexadezimal ohne führendes "0x". Eine Beispielobjektbeschreibungdatei ist `lib/ds301.txt` mit Beschreibungen der Objekte aus dem Kommunikationsprofil. Beim Laden einer EDS-Datei wird geprüft, ob in dem jeweiligen Verzeichnis eine Datei mit dem gleichen Namen aber mit der Dateiendung `.txt` vorhanden ist. Falls ja, so wird diese Datei als Beschreibungdatei für die EDS-Datei geladen. Ist keine Objektbeschreibungdatei für ein Gerät vorhanden, so wird im Objektbeschreibungsregister nur der Datentyp bzw. der Objektcode des Objekts dargestellt.

Das EtherCAT® DesignTool von *port* erzeugt die Objektbeschreibungdatei automatisch.

9.

9.1.

Der Objekt-Overview-Tab ermöglicht das gleichzeitige Lesen und Schreiben aller Subindizes eines Arrays oder Records in einer Maske.



Sub	Name	Data type	Access	Value
000	number of entries	u8	ro	0x4
001	Vendor Id	u32	ro	0x34
002	Product Code	u32	ro	0x0
003	Revision number	u32	ro	0x0
004	Serial number	u32	ro	0x0

Bild 19: Index overview

Beim Schreiben oder Lesen der Werte gibt es einige Einschränkungen. So können keine Subindizes mit dem Datentyp "Domain" gelesen oder geschrieben werden. Desweiteren ist das Schreiben von PDO- und PDO-Mapping aus dem Overview-Tab nicht möglich, da bei diesen Objekten speziellen Reihenfolgen beim Schreiben der Subindizes eingehalten werden müssen. Für diese Objekte kann die PDO-Konfigurationsmaske verwendet werden.

10.

Das PDO-Tab dient der einfachen Konfiguration von PDOs. Bei der Auswahl eines PDO-Objekts im EDS-Baum wird die PDO-Maske mit den Werten aus dem EDS aktualisiert. Die PDO-Parameter können über die Maske einfach eingestellt werden. Neue Objekte können aus dem EDS-Baum per Drag&Drop in die Mapping- Tabelle gezogen werden und innerhalb dieser Tabelle per Drag&Drop verschoben werden. Gelöscht werden Einträge innerhalb der Mapping-Tabelle per Doppelklick. Bei statischem Mapping ist die Tabelle unveränderbar.

PDO Configuration Frame

Object: 0x1601

Mapping Table

Index	Sub	Length	Name
0x3002	0x01	8	Ctrl
0x3002	0x02	8	Data Out 0
0x3002	0x03	8	Data Out 1
0x3002	0x04	8	Data Out 2
0x3002	0x05	8	Data Out 3
0x3002	0x06	8	Data Out 4

PDO Data

32

60

61

62

63

73

Read from EDS

Send PDO

Store to DCF data

Read from Device

Show PDO in Chart

Send to Device

Bild 20: PDO Tab

Nach einer Änderung des PDO-Mappings oder des PDO-Assignments ist die Netzwerk-PDO-Konfiguration über das Symbol "Reload Network Configuration" in der Toolbar zu aktualisieren, um die geänderten PDOs nutzen zu können.

Nachfolgende Tabelle beschreibt die Schaltflächen im unteren Teil der Maske.

Schaltfläche	Erläuterung
Read from EDS	Liest die Werte aus der EDS-Datei
Read from Device	Liest die aktuellen Werte vom Gerät
Send to Device	Überträgt die eingegebenen Werte zum Gerät
Store to DCF data	Speichert die eingegebenen Werte in den DCF-Datenpuffer
Send PDO	Sendet das aktuelle PDO zum Gerät. Die Werte des PDOs werden aus den Eingabefeldern oberhalb gelesen. Beachten Sie, dass PDOs nur im Zustand OPERATIONAL gesendet/empfangen werden können. Nur bei RPDOs des Geräts.
Show PDO in Chart	Richtet eine PDO-Indication-Funktion ein, so dass die Werte dieses PDOs im Chart-Tab angezeigt werden. Zuvor muss das Gerät mit Send to Device entsprechend konfiguriert sein. Beachten Sie, dass PDOs nur im Zustand OPERATIONAL vom Gerät gesendet werden können. Nur bei TPDOs des Geräts.

10.1.

Nach Betätigung der Schaltfläche "Show in Chart" erscheint folgender Dialog.

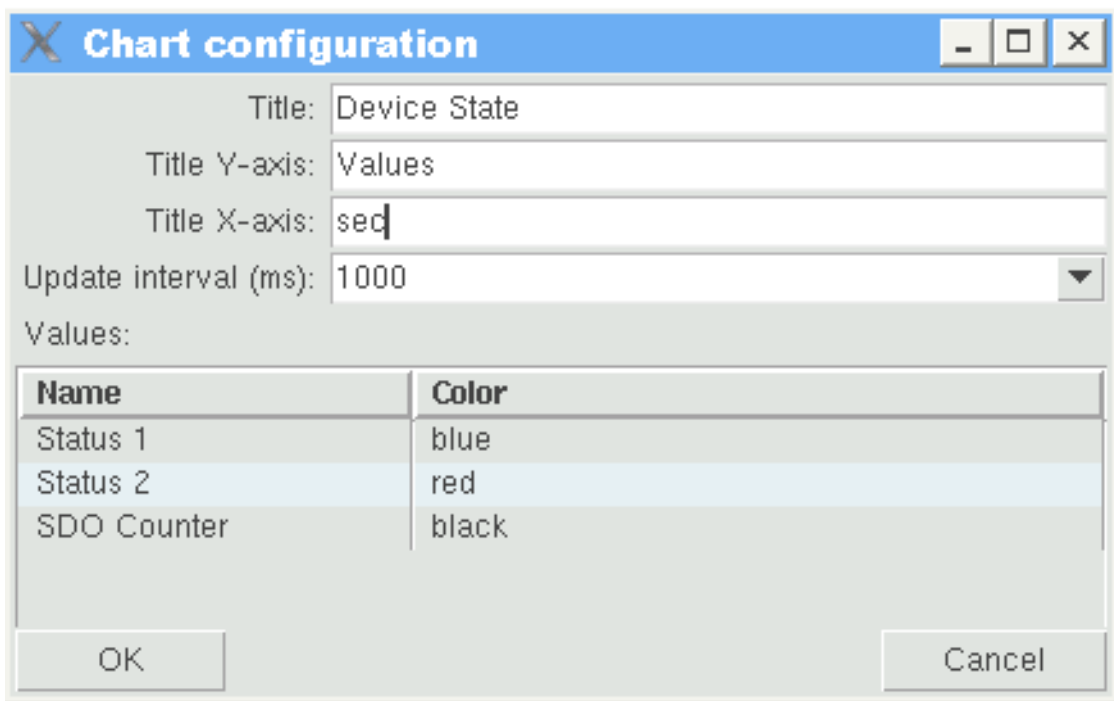


Bild 21: Chartkonfigurationsdialog

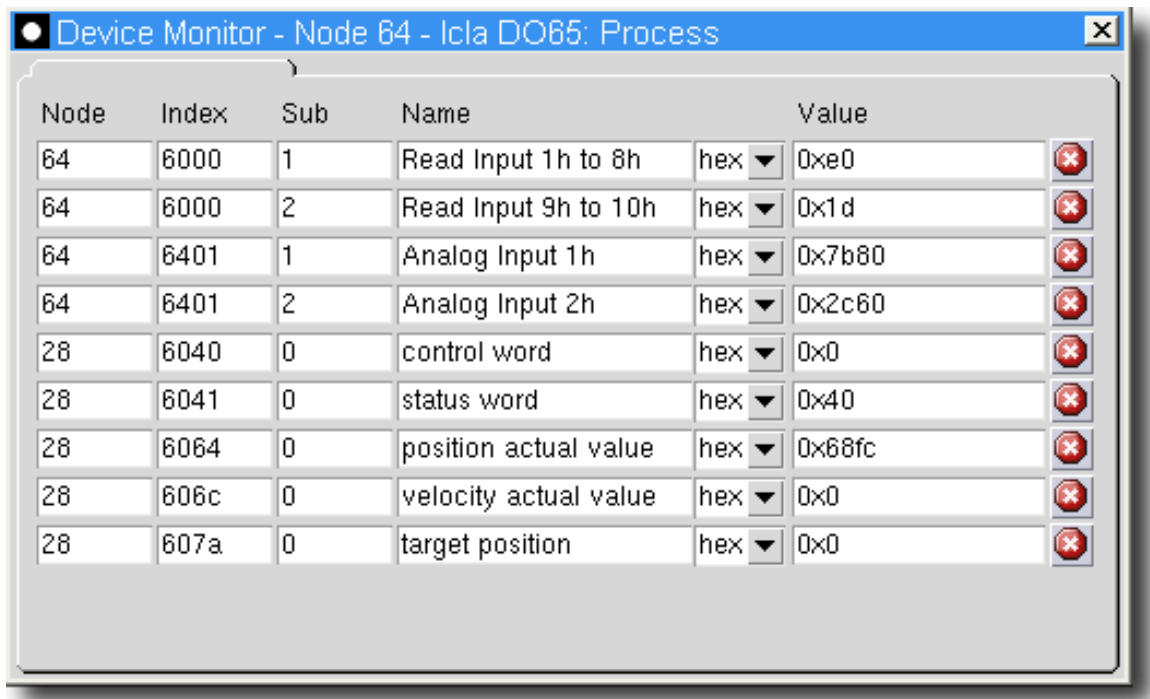
Die Beschriftungen des Diagramms, der Achsen und der einzelnen Daten können in diesem Dialog konfiguriert werden.

Wird ein **Updateintervall** ungleich 0 gewählt, so wird das Diagramm zeitgesteuert aktualisiert. D.h. Ankommende Daten werden in einem Buffer geschrieben und nach dem Ablauf eines bestimmten Intervalls wird der Wert des Buffers im Chart angezeigt. Bei einem Updateintervall von 0, wird der Chart bei jedem ankommenden PDO aktualisiert. Bei asynchronen PDOs geht dabei die Zeitinformation verloren.

Unter **Color** wird die Farbe eines Signals konfiguriert. Dabei ist jede von Tcl/Tk unterstützte Farbbezeichnung möglich. Siehe z.B. "man colors" unter Linux™. Zusätzlich können die Farben auch in hexadezimaler Notation angegeben werden (Bsp.: #ffaa11).

11.

Das Prozessabbild-Tab ermöglicht die Beobachtung von Prozessgrößen von verschiedenen Knoten. Die Werte der einzelnen Objekte werden zyklisch per SDO abgefragt und aktualisiert. Das Intervall beträgt per default 1000 ms. Im Standardfall wird die Anzeige nur aktualisiert, wenn das Prozessabbild-Tab im Vordergrund ist. Ist jedoch *Update even in background* aktiv, so werden die Werte in jedem Zyklus gelesen.



Node	Index	Sub	Name	Value
64	6000	1	Read Input 1h to 8h	hex 0xe0
64	6000	2	Read Input 9h to 10h	hex 0x1d
64	6401	1	Analog Input 1h	hex 0x7b80
64	6401	2	Analog Input 2h	hex 0x2c60
28	6040	0	control word	hex 0x0
28	6041	0	status word	hex 0x40
28	6064	0	position actual value	hex 0x68fc
28	606c	0	velocity actual value	hex 0x0
28	607a	0	target position	hex 0x0

Bild 23: Prozessabbild

Das Hinzufügen einzelner Objekte geschieht per Drag&Drop von EDS-Baum aus. Zum Löschen dient die kleine Schaltfläche neben dem Wert des Objekts. Ganze Arrays oder Records von mit einen Mal in das Prozessabbild eingefügt werden, indem der Hauptindex per Drag & Drop eingefügt wird. Die Art der Darstellung, hexadezimal, dezimal oder binär, kann in der Auswahlbox vor dem Wert eingestellt werden. Bei Strings hat diese Auswahl keine Auswirkung.

Beim Speichern des kompletten Projektes mit "File → Project → Save Project" werden die aktuellen Einstellungen gespeichert und beim Import des Projektes wieder geladen.

12.

Als Ergänzung zu dem SDO-basierenden Process-Tab zeigt der PDO-Process-Tab PDOs mit deren Prozessdaten an.

Sende-PDOs der EtherCAT®-Knoten können auf 2 Wegen zu dem PDO-Process-Image hinzugefügt werden:

- Drag&Drop mit dem PDO-Objekt aus dem Objektbaum und
- via "Add to PDO Image" im PDO-Konfigurationstab.

Das PDO-Process-Tab kann über die rechte Maustaste und **Clear PDO Process Image** gelöscht werden. Dies setzt auch die Einstellungen im EtherCAT®-Server zurück.

Die Ansicht wird, wenn das PDO-Process-Image-Tab aktiv ist, alle 1000 ms aktualisiert. Ist jedoch *Update even in background* aktiv, so werden die Werte auch aktualisiert, wenn das Tab nicht sichtbar ist.

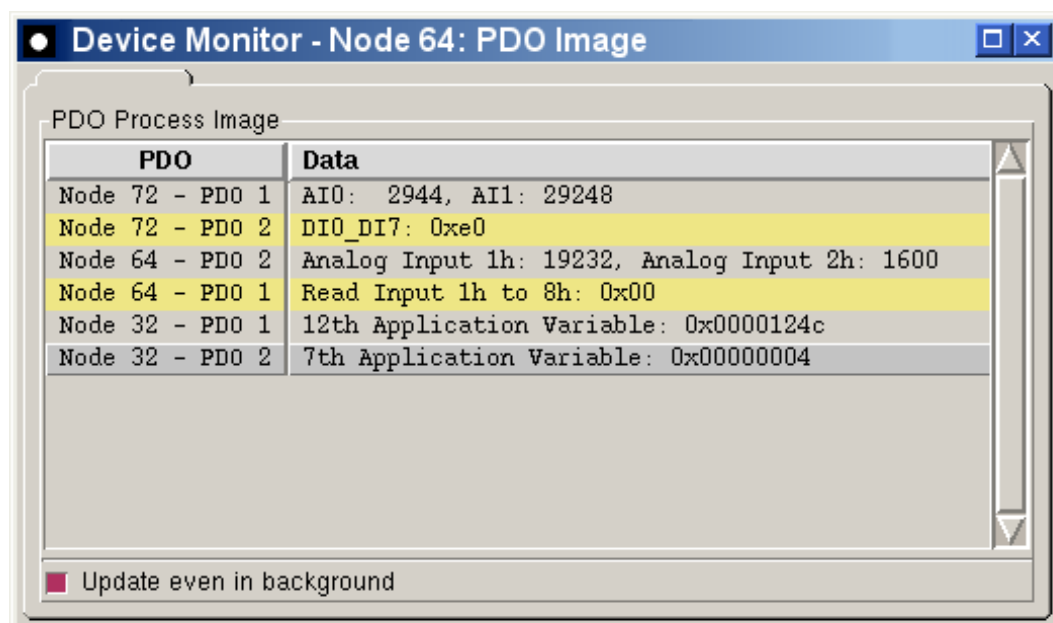


Bild 24: PDO Prozessabbild

Beim Speichern des kompletten Projekts mit **File → Project → Save Project** werden die aktuellen Einstellungen gespeichert und beim Import des Projektes wieder geladen.

13.

Mithilfe des Dienstes "File over EtherCAT (FoE)" können Dateien beliebiger Größe zu EtherCAT-Geräten übertragen oder von ihnen gelesen werden. Da jedoch nicht alle EtherCAT-Geräte diesen Dienst unterstützen, ist der FoE-Tab nur aktiv, wenn das aktive Geräte diesen Dienst auch unterstützt.

The screenshot shows the 'FoE' tab in the EtherCAT Device Monitor. It contains two main sections: 'Read File from Device' and 'Write File to Device'. Each section has three input fields: 'File Name on Device', 'Local File Name', and 'FoE Password'. The 'Read File from Device' section shows 'logging.txt' as the device file name, 'C:\projects\2009\404board\logging_20091002.txt' as the local file name, and '08150002' as the password. The 'Write File to Device' section shows 'linerization_tables.bin' as the device file name, 'C:\projects\2009\404board\linerization_tables.bin' as the local file name, and '08150001' as the password. Below each set of inputs is a button labeled 'Read File from Device' and 'Write File to Device' respectively. The top of the window has tabs for 'Action', 'NMT', 'Description', 'PDO Image', 'FoE', and 'ESC', with 'FoE' currently selected.

Bild 26: FoE Tab

Der FoE-Tab besteht aus 2 Teilen. Der obere Teil steuert das Lesen von Dateien vom Gerät und der untere das Schreiben zum Gerät. Für beide Vorgänge muss jeweils ein lokaler Dateiname auf dem PC, eine Dateiname auf dem Gerät und ein Passwort angegeben werden. Das Passwort muss dabei ein 32-bit Hexwert wie z.B. 0123ABCD sein.

Der Dateitransfer kann bei großen Dateien eine merkliche Zeit dauern. Daher wird der Fortschritt mit einem Fortschrittsbalken angezeigt und der EtherCAT® Device Monitor ist während des Vorgangs gesperrt.

Um die Dienste aus Skripten auszuführen, stehen die Funktionen `::foe::readFile deviceFileName password localFileName` und `::foe::writeFile <deviceFileName> <password> <localFileName>` zur Verfügung.

14.

Der EtherCAT® Device Monitor unterstützt das Lesen und Schreiben von Registern des EtherCAT Slave Controllers(ESC). Nach der Eingabe einer Startadresse werden nach Betätigung des "Read ESC"-Buttons 512 Byte des ESC ausgelesen und dargestellt.

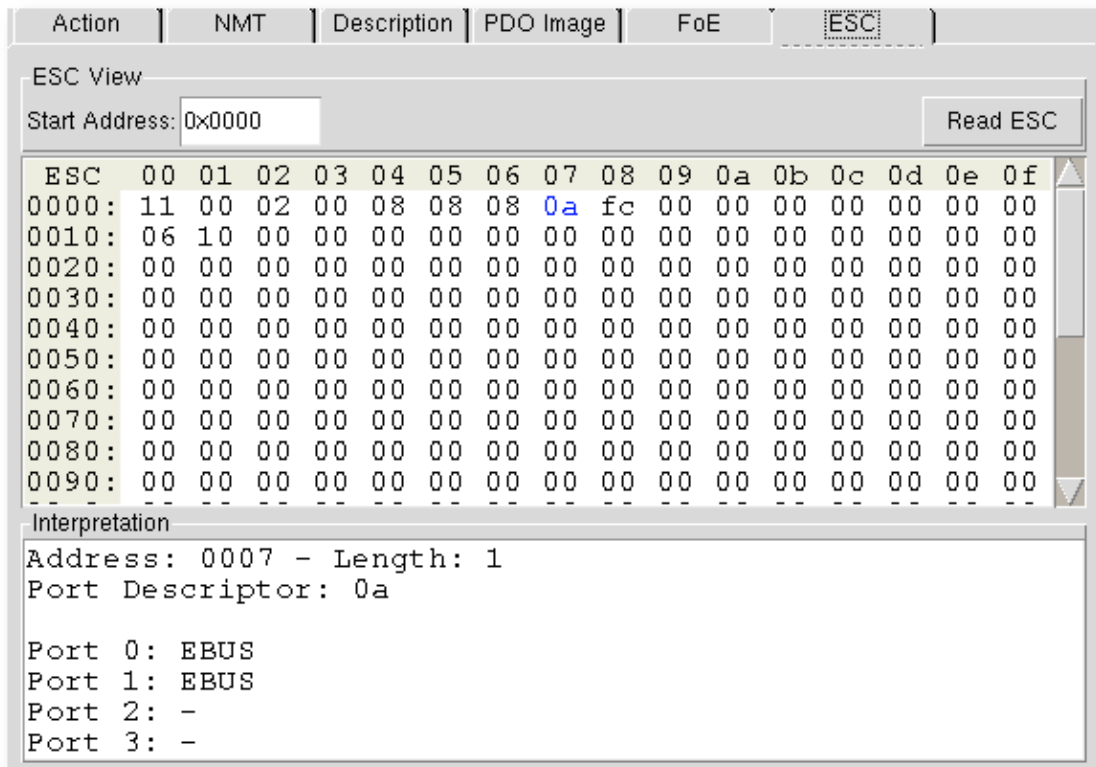


Bild 27: ESC Tab

Wird ein oder mehrere zusammengehörige Bytes in dem Registerabbild markiert, so wird im unteren Feld das Byte entsprechend seiner Bedeutung interpretiert.

Auch aus Skripten kann auf die ESC-Register mit den Funktionen `::esc::regRead <addr>` und `::esc::regWrite <addr> <value>` lesend und schreiben zugegriffen werden.

15.

Der EtherCAT® Device Monitor unterstützt das Lesen und Schreiben von Daten des EEPROMs. Nach der Eingabe einer Startadresse werden nach Betätigung des "Read EEPROM"-Buttons 512 Words des EEPROM ausgelesen und dargestellt.

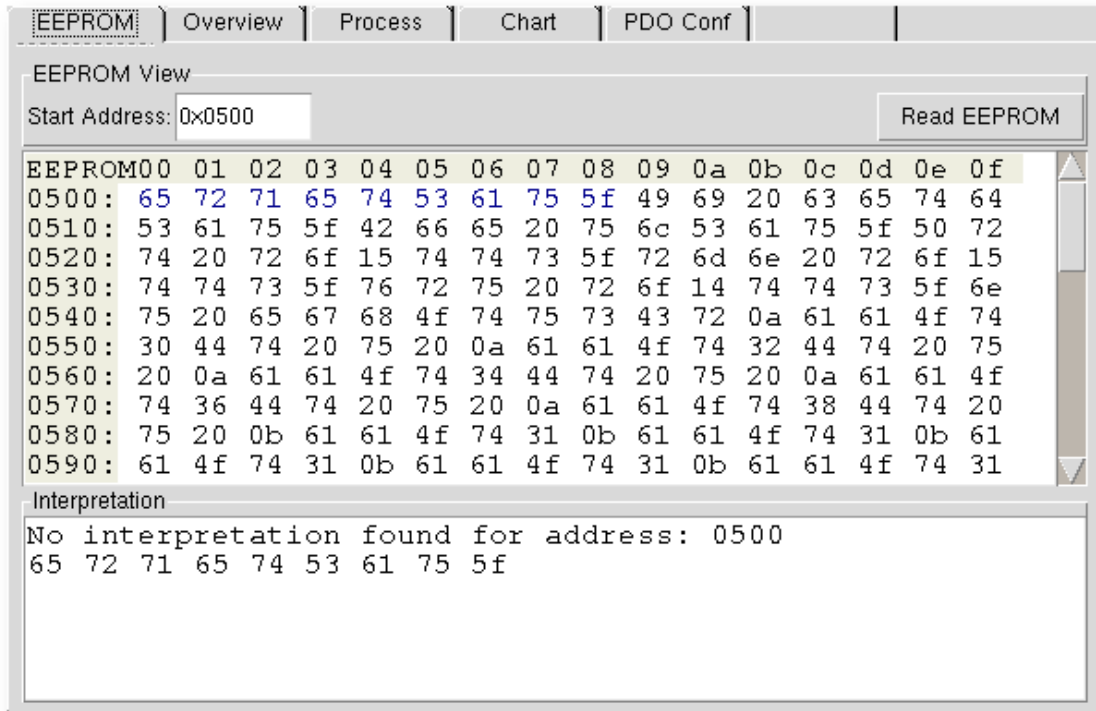


Bild 28: EEPROM Tab

Auch aus Skripten kann auf den EEPROM mit den Funktionen `::eeprom::siiRead <addr>` und `::eeprom::siWrite <addr> <value>` lesend und schreiben zugegriffen werden.

16.

Zur grafischen Daten-Visualisierung steht ein Linienschreiber zur Verfügung. Die Verwendung eines Linienschreibers zeigt beispielhaft die vorinstallierte Registerkarte **Chart**. Öffnet man die **Chart**-Registerkarte, sieht man einen vorbereiteten Linienschreiber. Neben der Verwendung zur Visualisierung von PDO-Daten kann der Chart auch doch Anwenderskripte benutzt werden. Die Wertübergabe erfolgt durch Aufruf der Prozedur `::edm::addChartData`. Die Prozedur ist folgendermassen definiert

```
proc ::edm::addChartData { valueList } {  
    # add new values to the strip-chart window  
}
```

In `valueList` werden die Werte für die Linienschreiber übergeben. Das Argument muss eine Liste mit Werten sein:

```
(EDS) 9 % ::edm::addChartData {1 2 3 }  
(EDS) 10 % ::edm::addChartData {-1 -2 -3}
```

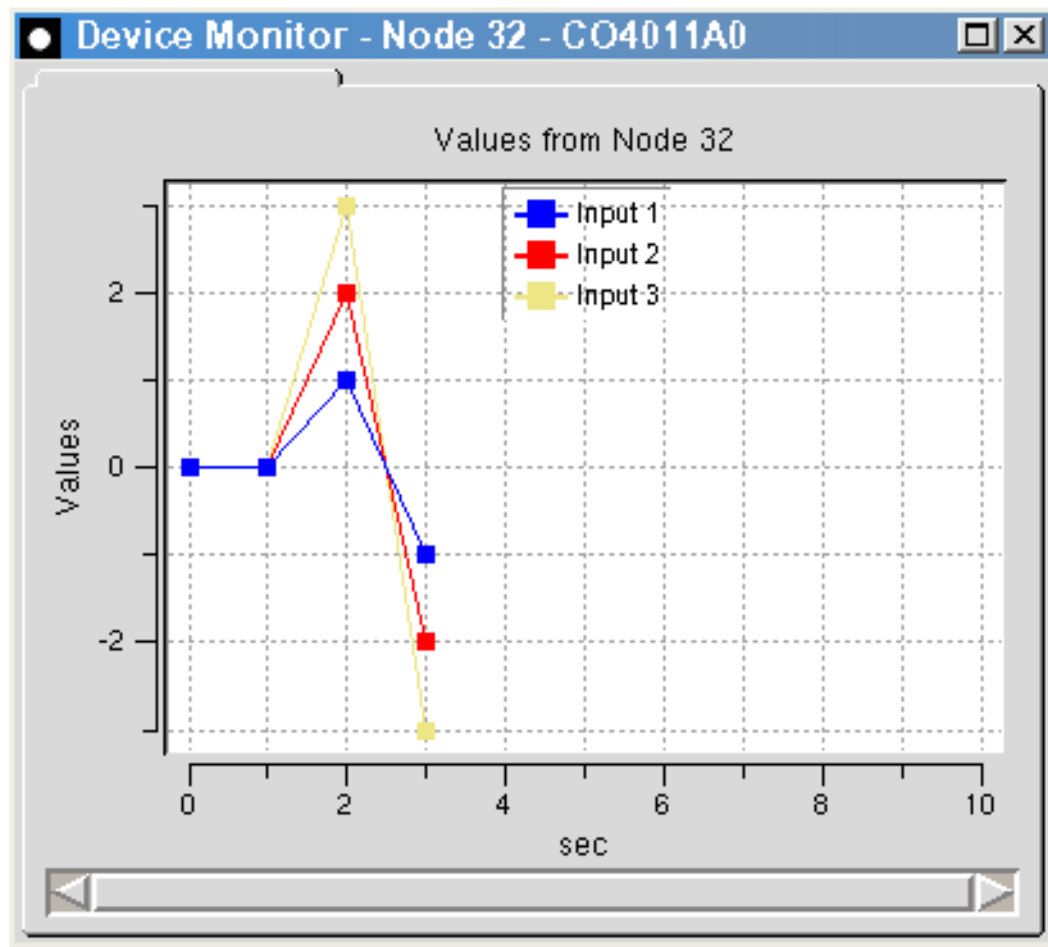


Bild 29: Linienschreiber nach zwei Aufrufen von `::edm::addChartData`

Mit geringem Aufwand lässt sich z.B. ein einfaches Skript in der Console schreiben, welches einen Werte von einem Gerät ausliest und im Chart visualisiert.

```
% proc readAndShow { index sub dataType } {
    set value [r $index $sub $dataType]
    edm::addChartData [list $value 0 0 0]
}
%
% # Call this procedure every second
% ::common::every "readAndShow 0x6401 0x01 i16" 1000
%
```

Windows -> Clear All , `::edm::clearChart` oder F7 löscht den Inhalt des Charts und mit der <Leertaste> kann die Aktualisierung der Anzeige angehalten werden.

Das Konsolen/Script-Kommando `::edm::saveChart <fileName>` speichert den Inhalt des Charts in einer Datei im Postscript-Format.

Mit der linken bzw. rechten Maustaste können Ausschnitte des Charts vergrößert bzw. verkleinert werden.

Über das Contextmenü über die rechte Maustaste sind folgende Aktionen möglich:

Menüeintrag	Bedeutung
Clear Chart	Entfernt alle Werte aus dem Chart.
Save Chart	Speichert das aktuelle Bild im Chart als PostScript-Datei.
Save Data	Speichert alle Werte im Chart als CSV-Datei.
Stop Updating	Stoppt die Aktualisierung des Charts

17.**17.1.**

Optional kann die Registerkarte Actions um weitere Elemente ergänzt werden. Eine vorbereitete Möglichkeit ist der Schieberegler (Bild 30).

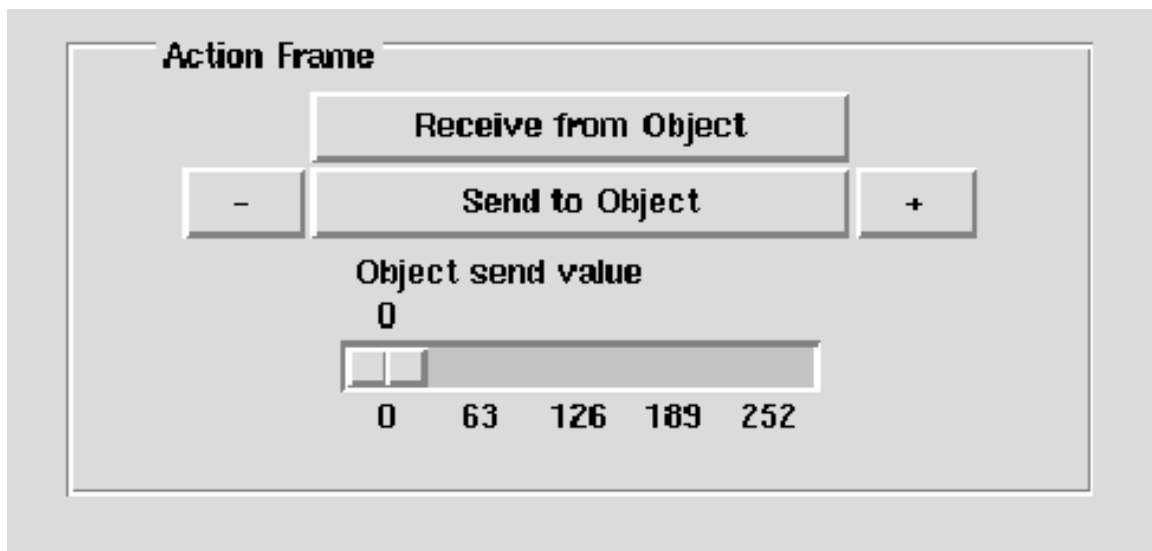


Bild 30, Registerkarte Action mit Schieberegler

Ein Schieberegler setzt sich aus einem Regler, der bewegt werden kann, und einer Skala zusammen. Zur Bedienung des Schiebereglers stehen folgende Tasten zur Verfügung:

- Durch einmalige, kurze Betätigung der linken Maustaste im entsprechenden Bereich des Schiebereglers, kann der zu sendende Wert um 1 erhöht bzw. verringert werden. Wenn die Maustaste gedrückt bleibt, wiederholen sich automatisch die Inkrementierungen bzw. Dekrementierungen.
- Wird die linke Maustaste über dem Regler betätigt, dann können die Werte durch Ziehen der Maus verändert werden.
- Wird die linke Maustaste mit gedrückter *Strg* -Taste im Trog gedrückt, dann bewegt sich der Schieberegler an den Endbereich in Richtung des Mauszeigers.
- Die 'hoch' und 'links' Cursorstasten bewegen den Schieberegler um 1 nach links.
- Die 'runter' und 'rechts' Cursorstasten bewegen den Schieberegler um 1 nach rechts.

Weiterhin ist eine Speicherung der definierten Objektwerte durch die Option *used for saving configuration* möglich.

17.2.

Ein Schieberegler kann für ein Objekt in einem separatem Fenster angezeigt werden. Dieses Fenster bleibt bestehen, auch wenn andere Objekte in der Baumstruktur angewählt werden.

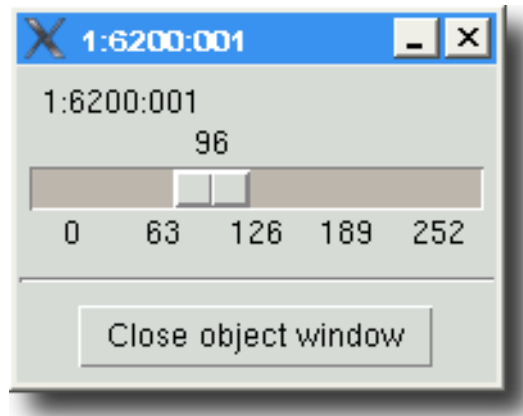


Bild 31: Schieberegler in separatem Fenster

Die möglichen Bedienaktionen für einen Schieberegler sind die gleichen, wie für den in Bild 30 beschrieben.

Zur Bedienung des Schiebereglers stehen folgende Bedienelemente zur Verfügung:

Close object window

Das Fenster mit dem Schieberegler wird geschlossen.

17.3.

Jedes numerische Objekt kann in einem separatem Fenster bitweise dargestellt werden.

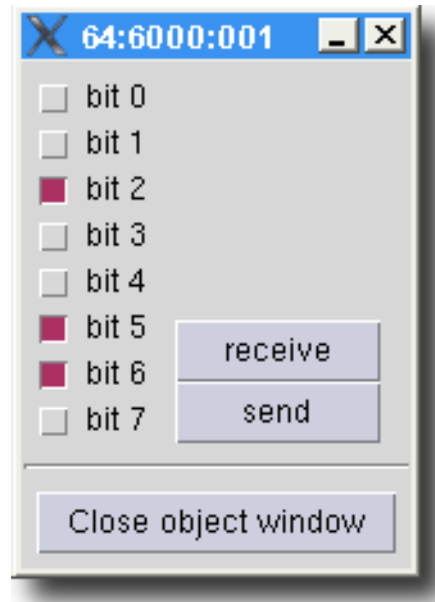


Bild 32: Bitdarstellung in separatem Fenster

Alle Objekterweiterungen können über die rechte Maustaste im Objektbaum den Objekten zugewiesen werden.

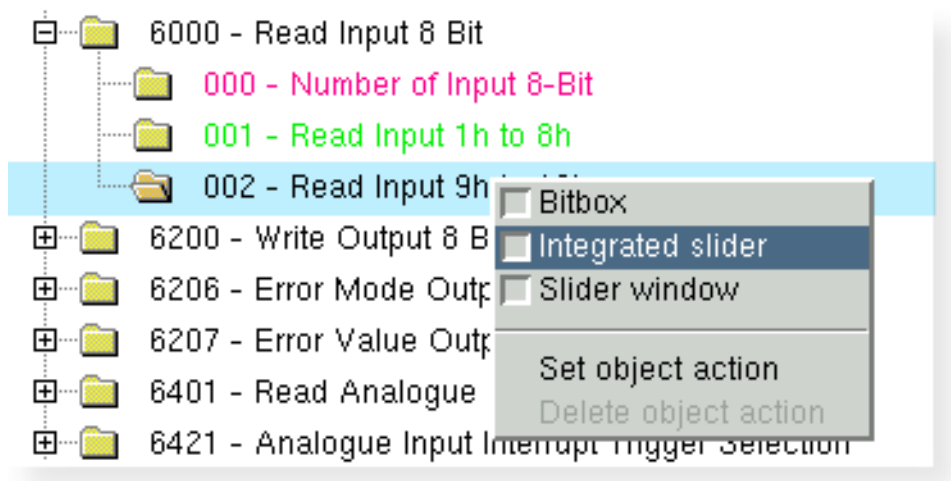


Bild 33: Zuweisung der Bedienerweiterung

Beim Speichern der Konfiguration in einer OCF-Datei bleiben alle Zuordnungen von Bedienerweiterung zu Objekten erhalten.

18.

Zusätzlich zu den vorhandenen Registerkarten kann jeder Nutzer eigene Registerkarten dem EtherCAT® Device Monitor hinzufügen.

Dabei wird zwischen 2 verschiedenen Arten von Registerkarten unterschieden:

- Registerkarten mit Testunterstützung
- Registerkarten ohne Inhalt

18.1.

Registerkarten mit Testunterstützung enthalten eine Liste von Skripts. Über ein Kontrollkästchen sind Skripts aus der Liste auswähl- und ausführbar.

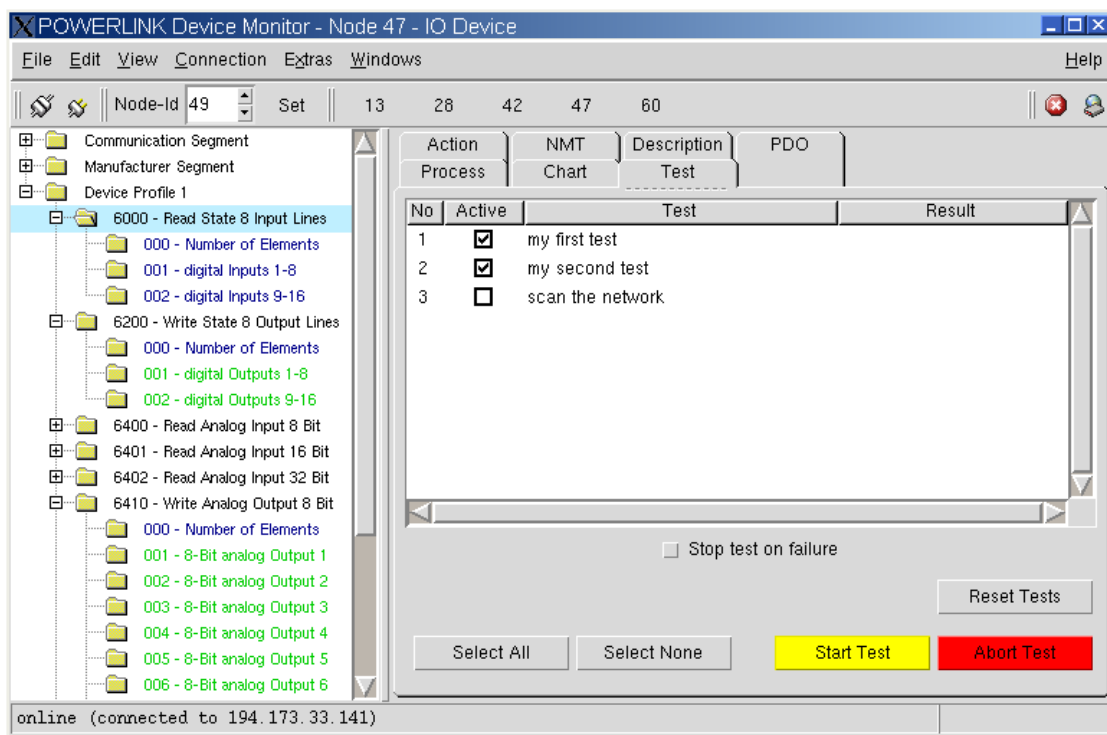


Bild 29: Registerkarte mit Testunterstützung

Eine Beispielkonfiguration wird im Verzeichnis "<Arbeitsverzeichnis>\demo_t\" mitgeliefert.

Die Konfiguration der Registerkarten erfolgt durch die Dateien

- "<Arbeitsverzeichnis>\demo_t\t_start.tcl" und
- "<Arbeitsverzeichnis>\demo_t\t_<conf1..n>.tcl".

In der Datei "<Arbeitsverzeichnis>\demo_t\t_start.tcl" werden alle zusätzlichen Registerkarten definiert. Der Inhalt einer einzelnen Registerkarte wird über die Datei "<Arbeitsverzeichnis>\demo_t\t_<conf1..n>" festgelegt, d.h. für jede zusätzliche Registerkarte

existiert eine Datei "<Arbeitsverzeichnis>\demo_t\t_<conf1..n>".

```
edm::addTestTab t_conf {<tab_name>}
```

Beschreibung:

erzeugt eine zusätzliche Registerkarte mit Testunterstützung im EDM

Eingabe:

t_conf	Name der Datei, die die Skriptliste für diese Registerkarte enthält, standardmäßig <code>t_files</code>
tab_name	Text für die Beschriftung der Registerkarte

Ausgabe:

interner Name der Registerkarte

Format von "<Arbeitsverzeichnis>t_<conf1..n>":

Zeilen, die mit '#' beginnen, sind Kommentare und werden übersprungen. Jede Zeile enthält eine Beschreibung eines Tests. Für einen Test sind folgende Angaben zu machen:

```
<script>.tcl      {<label>}    {Tcl procedure name}
```

script.tcl	Name der Datei, die ausgeführt werden soll
label	Text, der hinter dem Kontrollkästchen stehen soll
Tcl-Prozedurname	Tcl Prozedur, die den Test durchführt. Wenn der Test ohne Fehler durchgeführt wurde, muss 0 zurückgegeben werden. Jeder andere Rückgabewert bedeutet, dass der Test fehlerhaft war. Der Prozedur wird ein Parameter übergeben. Dieser enthält die Testbeschreibung aus <code>t_files</code> .

Der Name des Tcl-Skripts, der Text und die Tcl-Prozedur sind durch ein Leerzeichen oder einen Tabulator voneinander getrennt.

Zusätzlich können Tcl-Prozeduren registriert werden, die beim Start eines Testlaufs, Abbruch eines testlaufs, Fehler eines Tests und Ende eines Testlaufs ausgeführt werden. Diese Prozedure werden wie folgt registriert. Sie werden nur für die Tests aufgerufen die in der Datei `t_files` definiert sind.

```
@start <Tcl Procedure>
@abort <Tcl Procedure>
@error <Tcl Procedure>
@end    <Tcl Procedure>
```

BEISPIEL:

Es sollen zwei zusätzliche Registerkarten erstellt werden. Die erste Registerkarte soll die Standardbezeichnung "Test" und die zweite Registerkarte die Bezeichnung "Some Tests" erhalten. Beiden Registerkarten werden Musterprogramme zugeordnet. Bild 29 stellt die zusätzlichen Registerkarten dar.

t_start.tcl:

```
# add the test selection frames
# use standard values for the file (t_files) and name (Test)
edm::addTestTab
# use user-defined values for the file (t_files2) and name (Some Tests)
edm::addTestTab t_files2 {Some Tests}

proc my_startHandler { args } {

    cdm::banner
}

proc my_stopHandler { args } {

    puts "\n\n\n\n\n-----"
    puts "      date / sign"
    puts "\n===== E N D =====\n"
}

t_files:

# the first set of test scripts using the standard filename
#
@start my_startHandler
@stop  my_stopHandler

t_myfirst.tcl    {my first test}    my_1st_test
t_mysec.tcl      {my second test}   my_2nd_test
# scanning the network for EtherCAT® devices
t_scan.tcl       {scan the network} scan_test

t_files2:

# all testfiles for the second set of tests
# comments allowed after #
#
# Here we do not specify start and stop handlers
#   so they won't be called.
t_scan.tcl       {scan the network} scan_test
t_myfirst.tcl    {my first test}    my_1st_test
t_mysec.tcl      {my second test}   my_2nd_test
t_mythird.tcl    {another}          my_3rd_test
```

t_myfirst.tcl

```
#
# Write Heartbeat Producer and check if value can be read back.
#
#
proc my_1st_test { args } {

    wwc 0x1017 0 u16 1000 OK
    if { $::global_stop == "" } {
        return "Canceled"
    }
    rrc 0x1017 0 u16 1000

    return 0
}
```

Die Registerkarten werden angezeigt, nachdem File → Load File → t_start.tcl oder `source t_start.tcl` interaktiv in der Konsole ausgeführt wurde.

Änderungen an den Dateien "<Arbeitsverzeichnis>t_start.tcl" und "<Arbeitsverzeichnis>t_<conf1..n>" werden erst aktiv nach einem Neustart des EtherCAT® Device Monitors und einer Wiederholung des Ladevorgangs von "<Arbeitsverzeichnis>t_start.tcl".

Zur Bedienung stehen folgende Bedienfelder auf einer Registerkarte zur Verfügung:

Select All

Alle Skripts werden markiert.

Select None

Alle Markierungen werden gelöscht.

Start Test

Die Skripts werden der Reihe nach - zeilenweise - abgearbeitet.

Abort Test

Die Abarbeitung des/der Skripts wird abgebrochen.

Der Abbruch erfolgt spätestens nach der Beendigung des laufenden Skripts. Soll das laufende Skript selbst abgebrochen werden, muss das Skript die globale Variable *global_stop* auswerten und den Abbruch ausführen. Bei der Betätigung von "Abort Test" wird die globale Variable *global_stop* auf den Wert 1 gesetzt. Um den aktuellen Wert von *global_stop* zu erhalten, ist das Abfragen der Oberfläche nötig. Die Tcl-Funktion `update` führt diese Abfrage aus und aktualisiert die Variable *global_stop*.

BEISPIEL:

Programmrahmen für ein Testscript:

```
# global variables
global global_stop

# initialize global variables
set global_stop ""

# execute application
while { ($global_stop == "") } {
    .
    .
    # read actual value of global_stop
    # and update the GUI
    update
}
```

18.2.

Eine leere Registerkarte wird erzeugt durch das Kommando:

```
edm::addTab {<title> <pos>}
```

Beschreibung:

erzeugt eine zusätzliche leere Registerkarte

Eingabe:

title	Name der Registerkarte
pos	Stellung der Registerkarte in der Anzeige Vorbelegung: anfügen als letzte Registerkarte

Ausgabe:

interner Name der Registerkarte

Die leere Registerkarte kann frei gestaltet werden durch die Eingabe entsprechender Tcl-Kommandos über die Konsole oder per Tcl-Skript.

BEISPIEL:

```
#Eine leere Registerkarte an Stelle 0 mit dem Namen "Bedienung" anlegen.
#set name [edm::addTab Bedienung 0]

#In der Mitte am unteren Rand soll eine Bedientaste mit dem Namen "OK"
#erscheinen.
button $name.button -text "OK" -command {w 0x1017 0 u16 500}
pack $name.button -side bottom -anchor center
```

18.3.

Eine vorhandene Registerkarte kann gelöscht werden:

```
edm::deleteTab {<pos>}
```

Beschreibung:

löscht eine Registerkarte

Eingabe:

pos Stellung der Registerkarte in der Anzeige (beginnend mit 0)

Ausgabe:

keine

19.

Das die Konsole wird aktiviert über `View → Console`

In diesem Feld können Tcl Kommandos, vom Anwender programmierte Skripts und Prozeduren ausgeführt werden.

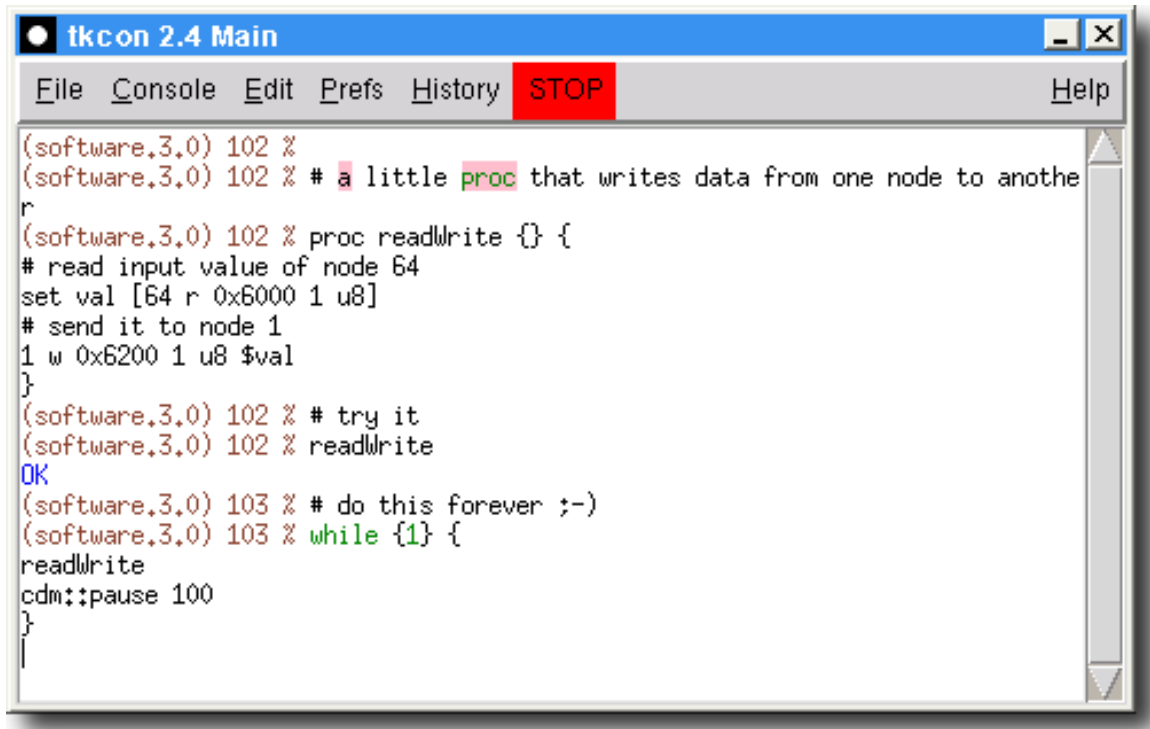


Bild 38: Konsole

19.1.

Eine Einführung in die Scriptsprache Tcl überschreitet den Rahmen dieses Handbuches. Zur Anschauung seien nur einige einfache Beispiele für das Arbeiten mit Tcl Kommandos in der Konsole angeführt.

Im Literaturverzeichnis befinden sich Hinweise auf Bücher und Webseiten zur Skriptsprache Tcl.

BEISPIEL:

```
set val 5           ;#setzt die Variable val auf den Wert 5
set val             ;#zeigt den aktuellen Wert der Variablen val an
puts "Hello"        ;#gibt das Wort Hello auf der Konsole aus
# Kommentar
set myarray(baud)   19200           ;#die Feldvariable myarray definieren
set myarray(parity) even
# Variablen werden durch Voranstellen eines "$" referenziert
puts "Bitrate: $myarray(baud)" ;#auf die Feldvariable myarray referenzieren
```

Weitere Informationen finden Sie auf www.tcl.tk (<http://www.tcl.tk>).

19.2.

Mit einem Texteditor lassen sich Abfolgen von Tcl Kommandos incl. von Kontrollstrukturen in Dateien speichern. Diese Dateien sind mit der Extension *.tcl zu versehen und können vom EtherCAT® Device Monitor geladen werden:

per Menü: File → Load File

interaktiv: in der Konsole `source <file>.tcl` eingeben

Die Kommandoabfolgen können dabei in Prozeduren zusammengefasst werden. Über den Namen der Prozedur ist die Kommandoabfolge ausführbar.

Erst nach dem Laden der Datei kennt der EtherCAT® Device Monitor alle in der Datei enthaltenen Prozeduren.

BEISPIEL:

Datei example.tcl:

```
# -----  
# show Hello  
# -----  
proc showHello { name } {  
    puts "Hello $name"  
    puts "How are you?"  
    return  
}
```

interaktiv in der Konsole laden:

```
source example.tcl
```

interaktiv in der Konsole ausführen:

```
$ showHello Heinz  
Hello Heinz  
How are you?
```

20.

Die DSP-402 Erweiterung besteht aus 3 Komponenten zur Vereinfachung der Arbeit mit DSP402-konformen Antrieben.

Diese Registerkarten werden über [Extras → Plug-in -> DSP 402 Extension](#) aktiviert. Sie stehen nur zur Verfügung, wenn die dazugehörige Lizenz erworben wurde.

20.1.

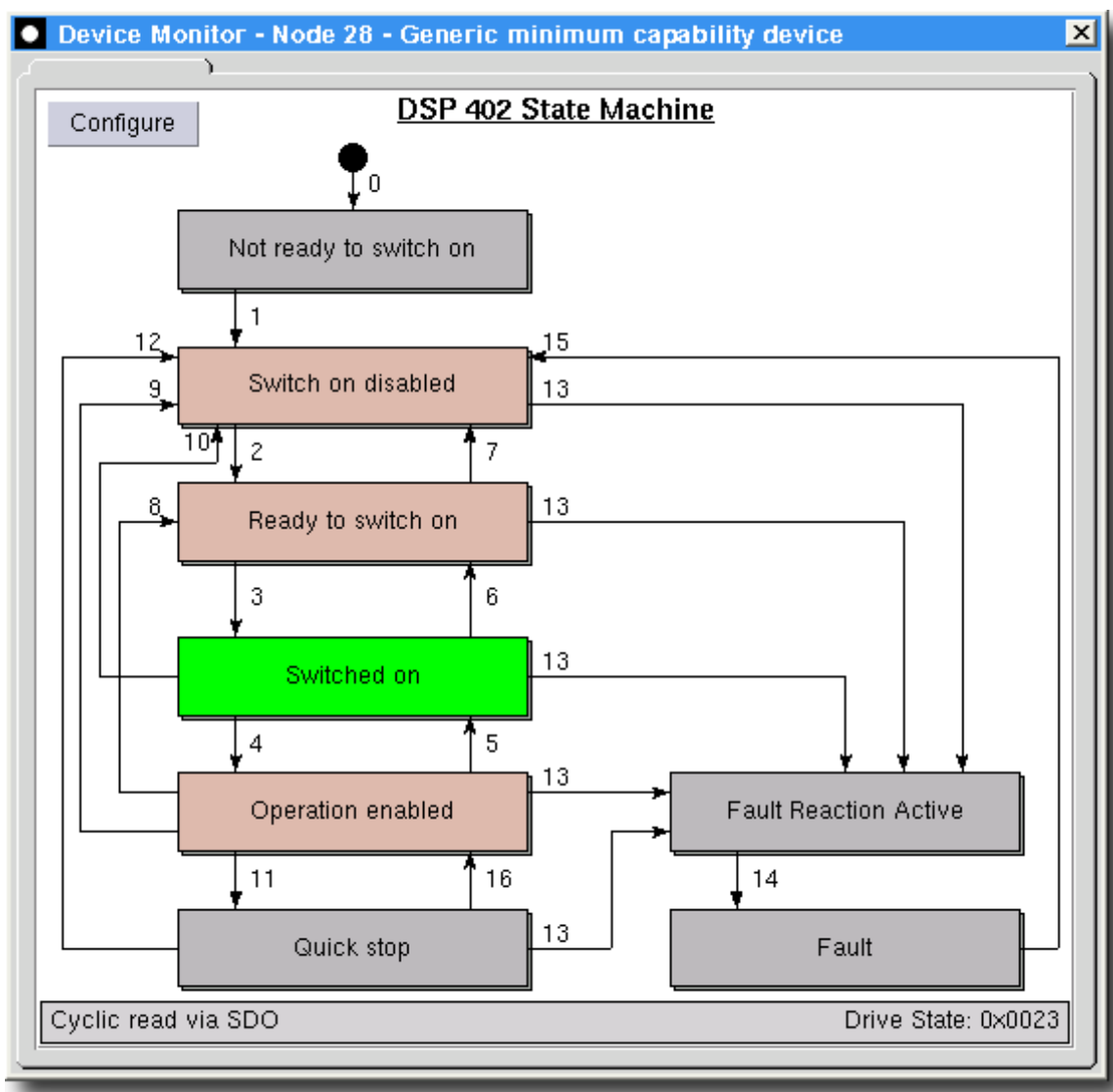


Bild 45: DSP402-Zustandsmaschine

Der grün hinterlegte Zustand ist der aktuelle Zustand des Antriebs. Hellbraune Felder kennzeichnen mögliche Folgezustände und graue Felder sind aus dem aktuellen Zustand nicht direkt erreichbar. Der aktuelle Wert des Statuswortes wird unten rechts unter der Zustandsmaschine angezeigt.

Über die Schaltfläche "Configure" öffnet sich der Konfigurationsdialog. Darin kann der Zugriffsmodus auf der Gerät und die zu verwendete Achse eingestellt werden. Die Einstellung der Achse gilt auch für die weiteren DSP402-Erweiterungen.

20.2.

Das Profile Velocity Mode Register vereinfacht die Steuerung von EtherCAT®-Antrieben im Profile Velocity Mode.

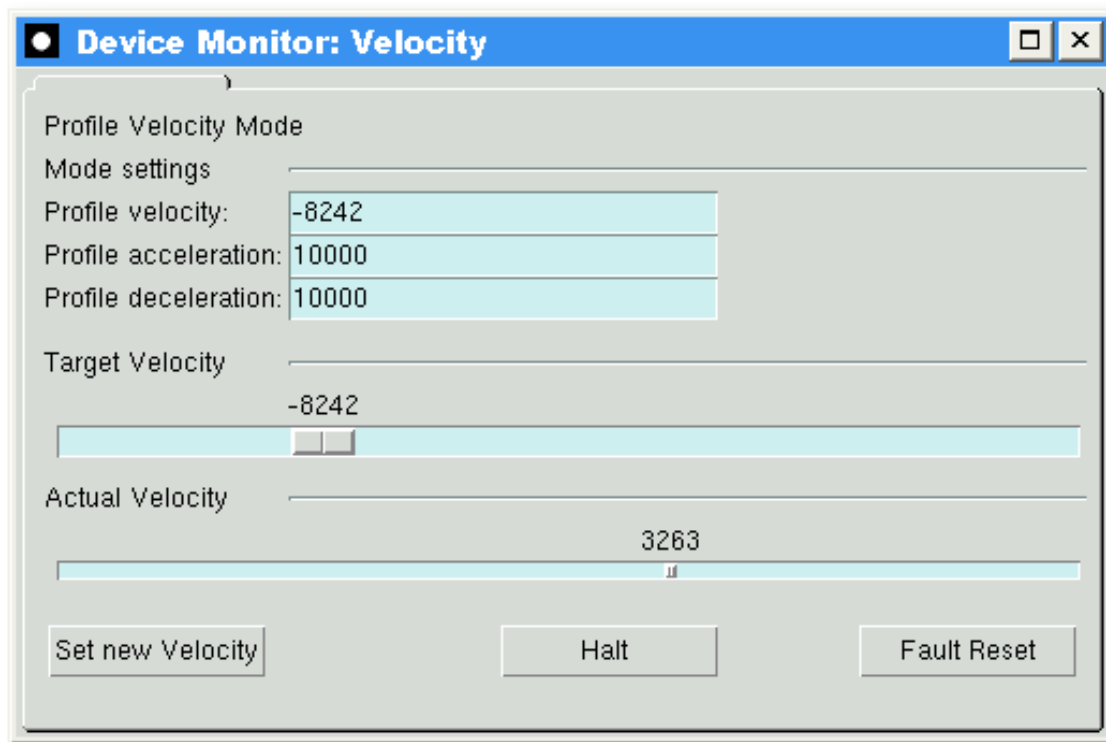


Bild 46: Profile Velocity Mode Register

20.3.

Das Position Mode Register vereinfacht den Test und die Inbetriebnahme von Positionierantrieben. Die Verfahrensgeschwindigkeit, die Beschleunigung und die Verzögerung sowie die Zielposition können komfortabel eingestellt werden. Grenzwerte für diese Parameter werden bei der Initialisierung des Registers aus dem entsprechenden Objekten des Gerätes gelesen und können nicht überschritten werden. Nach dem Betätigen der Start-Schaltfläche führt der Antrieb die vordefinierte Bewegung aus. Die Kommunikation mit dem Gerät erfolgt dabei via SDOs.

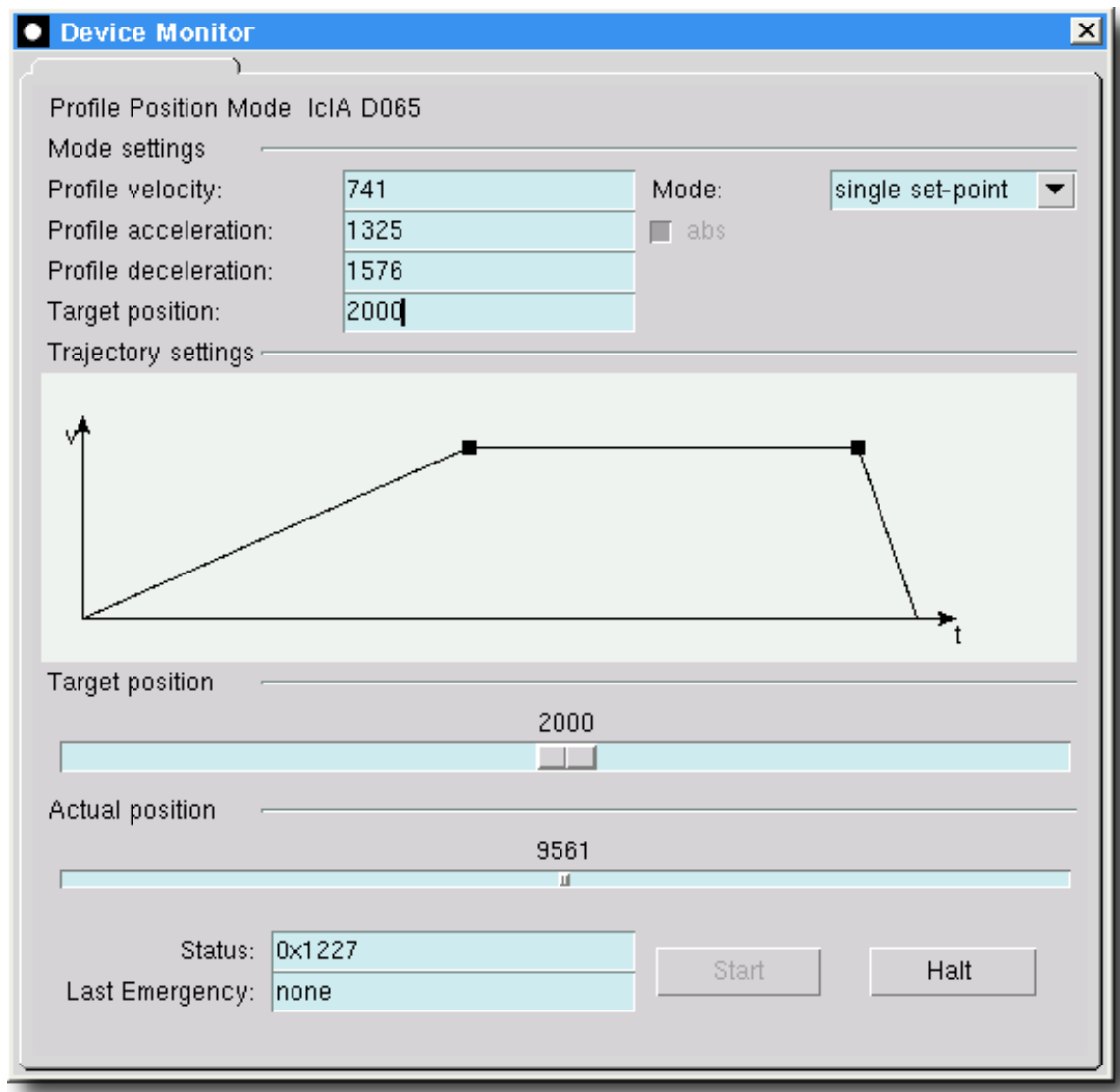


Bild 47: DSP402 Position Mode

20.4.

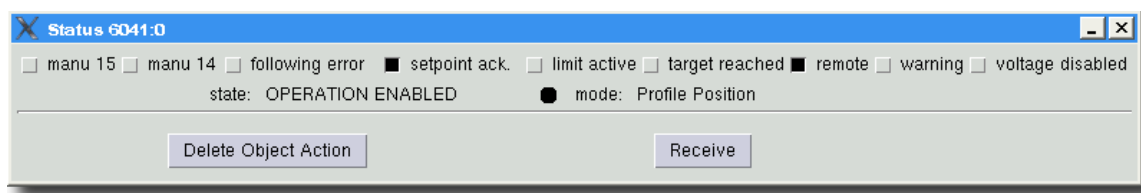


Bild 48: Status Word Bitbox

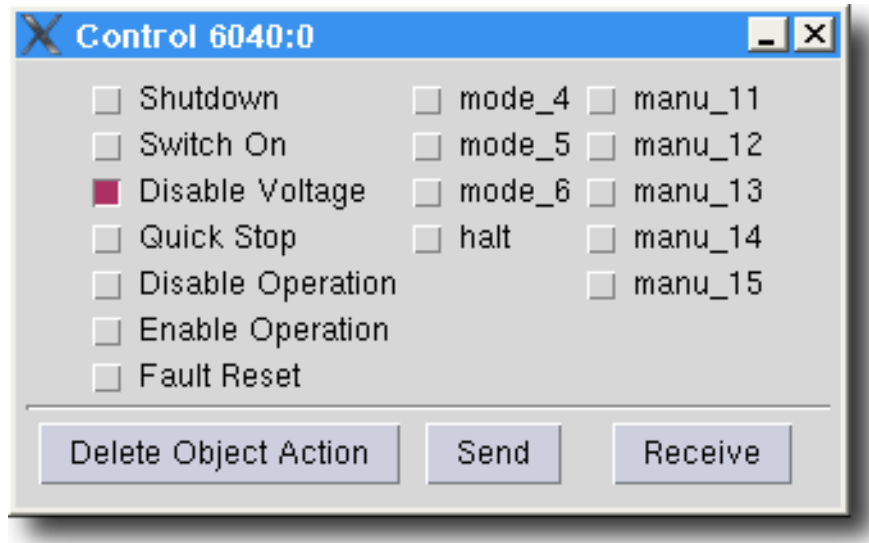


Bild 46: Control Word Bitbox

Diese Objekterweiterungen sind spezielle Bit-Boxen für das Status- und das Control-Word.

21.

21.1.

Im About-Dialog werden Informationen über:

- die aktuelle Version,
 - die Programmvariante,
 - den Lizenznehmer,
 - die Lizenz und
 - über die verfügbaren Tcl/Tk-Pakete.
- dargestellt.

21.2.

Für die Abfrage der Daten über die aktuellste Version des EtherCAT® Device Monitor wird eine http-Verbindung zu unserem Server hergestellt und die Daten (ca. 30 byte) werden heruntergeladen.

Dabei werden keine Daten (außer Ihrer IP-Adresse) zu dem Server gesandt.

22.

22.1.

r <index> <subindex> <typ>

Description:

SDO read procedure

Parameters:

index	object index
subindex	object subindex
typ	datatype of object <u8 u16 u32 i8 i16 i32 r32 vs>

Results:

read	value
------	-------

rr <index> <subindex> <typ>

Description:

SDO read procedure: type out the SDO read command and the answer from the device

Parameters:

index	object index
subindex	object subindex
typ	datatype of object

Results:

read	value
------	-------

rrc <index> <subindex> <typ> <ref>

Description:

SDO read and compare procedure: type out the SDO read command, type out the answer from the device and compare the received value with the reference value

In case of an error the global variable test_error is set to 1.

Parameters:

index	object index
subindex	object subindex
typ	datatype of object
ref	reference value

Results:

0	received value is equal to the reference value
1	received value differs from the reference value

rre <index> <subindex> <typ>

Description:

SDO read and message error: type out the SDO read command,
type out the answer from the device and check the received value

Parameters:

index	object index
subindex	object subindex
typ	datatype of object

Results:

0	received no SDO abort domain transfer
1	received SDO abort domain transfer

w <index> <subindex> <typ> <val>

Description:

SDO write procedure

Parameters:

index	object index
subindex	object subindex
typ	datatype of object <u8 u16 u32 i8 i16 i32 r32 vs>
val	value

Results:

OK	value has been written to the object
ERROR*	CiA 309-3 error code if SDO transfer failed

ww <index> <subindex> <typ> <val>

Description:

SDO write procedure: type out the SDO write command and the answer from the device

Parameters:

index	object index
subindex	object subindex
typ	datatype of object
val	value to write

Results:

nothing

wwc <index> <subindex> <typ> <val> <expected>

Description:

SDO write and compare the expected result
the returned value is compared with the expected one.
typical a write can return "OK" or some errors beginning with "error"

one or two lines with the command and the result are printed to stdout.

the last line contains a right justified flag for

OK - the returned value matches the expected

FAILURE - the returned value does not match the expected

In case of an error the global variable test_error is set to 1.

Parameters:

index	object index
subindex	object subindex
typ	datatype of object
val	value
expected	the expected return string

Results:

0	received value is equal to the expected value
1	received value differs from the expected value

wwe <index> <subindex> <typ> <val>

Description:

SDO write and message error: type out the SDO write command,
type out the answer from the device and check the received value

Parameters:

index	object index
subindex	object subindex
typ	datatype of object
val	value

Results:

0	received no SDO abort code
1	received SDO abort code

22.2.

`::cdm::addTab <titel> <pos>`

Description:

adds a tab into the Tabset of the right side

Parameters:

titel	name of the Tab displayed at the ...
pos	position starting with 0, can be end

Results:

frame	the window name of the top level frame within this tab
-------	--

`::cdm::deleteTab <pos>`

Description:

deletes a tab from the tabset

Parameters:

pos	position starting with 0, can be end
-----	--------------------------------------

Results:

`::cdm::addTestTabOld <filename> <title>`

Description:

adds a special tab on the right side of the device monitor;
the tab-card contains checkbuttons for test-scripts,
the list with scripts is in a file and will set by user

Parameters:

filename	name of the file which contains a list of scripts for the special tab
title	title of the tab

Results:

window	path to Tab
--------	-------------

`::cdm::stringCenter <string> <l>`

Description:

center string -- prepend spaces to a given string
if the result will be printed it looks like it is centered within
a line lenght of l

Parameters:

string	unformatted string
l	desired line length

Results:

centered	string
----------	--------

```
::cdm::stringFill <string> <endword> <l>
```

Description:

append spaces and endword at string until line length l

Parameters:

string	unformatted string
endword	optional END-word, defaults to {}
l	desired line length, defaults to 80

Results:

formatted	string
-----------	--------

```
::cdm::banner
```

Description:

prints a headline with device and user characteristic

Parameters:

nothing

Results:

nothing

```
::cdm::putsDateTime
```

Description:

prints the current date and time

Parameters:

nothing

Results:

nothing

```
::cdm::commentInput  <wtitle> <cancelstring>
```

Description:

user input for comments

Parameters:

wtitle	window title
cancelstring	cancel string

Results:

nothing

```
::cdm::userDialog  <title> <type>
```

Description:

user response dialog
if a dialaog is finished with "Not Ok" another dialogbox
for giving a reason is opened

Parameters:

title	additional text for displaying to the user
type	specifies the type of dialogue
-	0 ... only wait for OK
-	1 ... decide between OK and NotOk
-	2 ... decide between OK NotOk and Abort
-	3 ... decide between OK and NotOk without comment
-	4 decide between list of given choices.

Results:

0	decided for OK
1	decided for Not OK
2	decided for Abort
or	selected button(text) for type 4

22.3.

`::cdm::getObjectType <node> <index>`

Description:

Returns the objectType

Parameters:

node	node-ID ({} means current node ID) as %d or 0x%x
index	index as %X, or 0x%x

Results:

Returns	objectType (VARIABLE,ARRAY,RECORD,DOMAIN)
throws	an error, if the index does not exist

`::cdm::getDataType <node> <index> <sub>`

Description:

Returns the data type of a sub index

Parameters:

node	node-ID ({} means current node ID) as %d or 0x%x
index	index as %X, or 0x%x
sub	subindex (opt) as %03d, %0x or 0x%x

Results:

Returns	DataType (u8 u16 u32 i8 i16 u32 r32 vs)
throws	an error, if the index does not exist

`::cdm::getName <node> <index> <sub>`

Description:

Returns the parameter name of an object from the EDS

Parameters:

node	node-ID
index	index
sub	sub index (opt) as %03d, %0x or 0x%x

Results:

returns	the parameter name or throws an error if the object does not exist
---------	--

`::cdm::getDefaultValue <node> <index> <sub>`

Description:

Returns the default value of an object from the EDS

Parameters:

node	node-ID
index	index
sub	sub index (opt) as %03d, %0x or 0x%x

Results:

returns the default value, throws an error if the object does not exist

`::cdm::existObject <node> <index> <sub>`

Description:

Checks, if an object exists in the EDS file

Parameters:

node	node-ID ({} means current node ID) as %d or 0x%x
index	index as %X, or 0x%x
sub	subindex (opt) as %d, 0x%x

Results:

1	object exists
0	object does not exist

`::cdm::getRemoteID`

Description:

Returns the current remote ID

Parameters:

-

Results:

Returns the remote-id

`::cdm::setRemoteID <id>`

Description:

Sets the remote id and updates the OD tree

Parameters:

id remote node id

Results:

1 success
0 invalid ID

```
::cdm::loadEds    <fileName>
```

Description:

Loads an EDS file for a node and after that
it looks for a description file with a matching name
and loads it.
If there is a matching device-specific .rc-file
it is sourced to. This file must contain valid Tcl or CDM commands

Parameters:

fileName path to EDS file (POSIX style)

Results:

-

```
setMessageLogLimit    <limit>
```

Description:

Set a new limit of lines for the message log

Parameters:

limit max number of lines in message log

Results:

Returns OK if limit is ok, otherwise a verbose error

```
::cdm::hideGUI    <flag>
```

Description:

Hides the CDM GUI.
This function is useful for scripts that build their own
user interface.

Parameters:

flag (opt.) use -noconsole to exclude the console window

Results:

-

`::cdm::showGUI <flag>`

Description:

Shows the CDM GUI. Counterpart of `::cdm::hideGUI`
This function is useful for scripts that build their own
user interface.

Parameters:

flag (opt.) use `-noconsole` to exclude the console window

Results:

-

`isColor <color>`

Description:

Checks if a color is a valid tcl color

Parameters:

color	color name or #hex expression
-------	-------------------------------

Results:

1	is valid color
0	is no valid color

`int2asc <i>`

Description:

Converts an unsinged char value into a ASCII representation

Parameters:

i	unsigned char value
---	---------------------

Results:

Returns	an ascii value
---------	----------------

`int2bits <i> <digits>`

Description:

Converts an integer value into a binary representation like
0b01010101

Parameters:

i	integer value
digits	length of the returned value (opt.)

Results:

Returns	a binary value
---------	----------------

`::common::every <script> <ms>`

Description:

This proc runs a script cyclically.
The global variable every(script) stores the after-id
for each script.

Parameters:

script	script to run
ms	interval in ms (opt.) defaults to 1000

Results:

-

clear_message_log

Description:

This proc deletes the content of the message log

Parameters:

-

Results:

-

save_message_log <filePath>

Description:

This proc saves the content of the message log into a file

Parameters:

filePath	path to writable file
----------	-----------------------

Results:

-

clear

Description:

This proc deletes the content of the CDM console

Parameters:

-

Results:

-

tkcon save <filePath> <spec>

Description:

This command saves the content of the CDM console to a file

Parameters:

filePath	path to writeable file
spec	content specifier (use all for all content)

Results:

-

22.4.

::402::ret

Description:

List of possible return values

Parameters:

-

Results:

0	OK
1	Drive in wrong state
2	Transition not possible
3	SDO abort occurred
4	No setpoint acknowledge

::p402::ppHandleNewSetpoint <delayTime>

Description:

This function handles a new setpoint in the Profile Position mode.

The function initiates an absolute movement in the single-setpoint mode.

If this function returns with an error it is possible that the mode-specific bits in object 0x6041 (controlword) and object 0x6041 (statusword) are not cleared.

Each drive needs a specific time for the transfer of a new setpoint. This time can be specified by the argument delayTime.

The communication is done via SDO.

This function can only be used for single drive devices.

This function is used by other functions of this namespace.

Parameters:

delayTime time for transfer in ms

Results:

ret see variable ::p402::ret

::p402::getState

Description:

This function gets the actual CiA-402 state of the drive.

The actual CiA-402 state is returned

in the format of object 0x6041 (statusword).

The communication is done via SDO.

This function can only be used for single drive devices.

This function returns a list with the following elements: retsee variable ::p402::ret stateactual CiA-402 state in statusword format The state is only valid if this function returns with ok.

Example for usage:

```
set retList [::p402::getState]
set retVal [lindex $retList 0]
if { [set retVal] != [set ::p402::ret(OK)] } {
  puts "Error: getState() returns with [set retVal]."
}
set actualState [lindex $retList 1]
```

If actualState is 0x0027 the drive is in the state OPERATION ENABLED.

Parameters:

-

Results:

retList value of ::p402::ret + actual CiA-402 state

```
::p402::changeState   <state> <delayTime>
```

Description:

This function changes into the desired CiA-402 state.

The desired CiA-402 state must be input

in the format of object 0x6041 (statusword).

Each drive needs a specific time to change the CiA-402 state.

This time is set by the argument delayTime.

The communication is done via SDO.

This function can only be used for single drive devices.

Example: The argument state must be 0x0027 for a change into the CiA-402 state OPERATION ENABLED.

Parameters:

state	desired CiA-402 state
delayTime	maximal time for state changing in ms

Results:

ret	see variable ::p402::ret
-----	--------------------------

::p402::halt

Description:

This function activates the halt function, i.e. the motion is halted.
The Halt bit in object 0x6040 (controlword) is set.
To reset Halt use the function ::p402::<mode>Change, because the necessary functionality to reset Halt is mode-specific.
The communication is done via SDO.
This function can only be used for single drive devices.

Example for reset Halt:
use ::p402::ppChange for the pp mode

Parameters:

-

Results:

ret	see variable ::p402::ret
-----	--------------------------

::p402::modeStop <delayTime>

Description:

This function stops a motion by the CiA-402 state transition from the CiA-402 state OPERATION ENABLED into the CiA-402 state SWITCHED ON. The drive is stopped.
The communication is done via SDO.
This function can only be used for single drive devices.

Note: The operation mode is not changed to NO_MODE, because not all drives support NO_MODE.

Parameters:

delayTime	maximal time for state changing in ms
-----------	---------------------------------------

Results:

ret see variable ::p402::ret

::p402::pvStart <targetVelocity> <profileAcceleration>
<delayTime>

Description:

This function starts a motion in the Profile Velocity mode.
The Profile Velocity mode is configured by the mandatory objects of the pv-mode.
The operation mode is set to Profile Velocity.
The motion is started by the change into the CiA-402 state OPERATION ENABLED.
The communication is done via SDO.
This function can only be used for single drive devices.

Parameters:

targetVelocity	value of object 0x60FF
profileAcceleration	value of object 0x6083
delayTime	maximal time for state changing in ms

Results:

ret see variable ::p402::ret

::p402::pvChange <targetVelocity>

Description:

This function changes the velocity of the movement in the Profile Velocity mode.
The communication is done via SDO.
This function can only be used for single drive devices.

Parameters:

targetVelocity	value of object 0x60FF
----------------	------------------------

Results:

ret see variable ::p402::ret

::p402::ppStart <targetPos> <profileVel> <profileAcc>
<delayTime>

Description:

This function starts a motion in the Profile Position mode.
The Profile Position mode is configured by the mandatory objects of the pp-mode.
The operation mode is set to Profile Position.
This function initiates an absolute movement in the single-setpoint mode.
The motion is started by the change into the CiA-402 state OPERATION ENABLED and the execution of the new-setpoint handling.
Each drive needs a specific time for the transfer of a new setpoint. This time can be specified by the argument delayTime.
The communication is done via SDO.
This function can only be used for single drive devices.

Parameters:

targetPos	value of object 0x607A
profileVel	value of object 0x6081
profileAcc	value of object 0x6083
delayTime	transfer time in ms

Results:

ret	see variable ::p402::ret
-----	--------------------------

::p402::ppChange <targetPosition> <delayTime>

Description:

This function changes the target position in the Profile Position mode.
This function initiates an absolute movement in the single-setpoint mode.
The movement is started by the execution of the new-setpoint handling.
Each drive needs a specific time for the transfer of a new setpoint. This time can be specified by the argument delayTime.
The communication is done via SDO.
This function can only be used for single drive devices.

Parameters:

targetPosition	value of object 0x607A
delayTime	transfer time in ms

Results:

ret see variable ::p402::ret

Literaturverzeichnis

- [1] Tcl and the Tk Toolkit
Ousterhout, John K.
Addison-Wesley, 1994
ISBN 0-201-63337-X
- [2] Practical Programming in Tcl and Tk, 2d ed.
Welch, Brent
Prentice Hall, 1997
- [3] Tcl/Tk Tools
Harrison, Mark
O'Reilly & Associates, 1997
- [4] Effective Tcl/Tk Programming
Harrison, Mark; McLennan, Michael
Addison-Wesley, 1998

Literatur im Internet

- [5] <http://www.tcl.tk>
- [6] <http://wiki.tcl.tk>
- [7] <http://www.activestate.com/solutions/tcl/>
- [8] <http://incrtcl.sourceforge.net/blt/index.html>
The BLT Toolkit.
BLT ist eine Erweiterung des Tk Toolkits, welches über zusätzliche Widgets, Geometriemanager und Kommandos verfügt.

23.

CAN	Controller Area Network
CAL	CAN Application Layer (CANopen base)
CDM	CANopen Device Monitor
CiA	CAN in Automation international users and manufacturers group e.V.
CN	Controlled Node
COB	Communication Object (CAN Message)
COB-ID	Communication Object Identifier
CSDO	Client SDO
EDM	EtherCAT Device Monitor
EDS	Electronic Data Sheet
ESI	EtherCAT Slave Information file
EMCY	Emergency Object
EPG	EtherCAT POWERLINK Standardization Group
ETG	EtherCAT Technology Group
MN	Managing Node
NMT	Network Management
OD	Object Dictionary
PDO	Prozess-Daten-Objekt, unbestätigter Dienst für die Echtzeitkommunikation
RPDO	Receive PDO
RTR	Remote Transmission Request
PDM	POWERLINK Device Monitor
PRMS	Problem Report Management System
SDO	Service-Daten-Objekt, bestätigter Dienst für den Zugriff auf das Objektverzeichnis
SSDO	Server SDO
SYNC	Synchronization Object
Tcl	Tool Command Language (Skript-Sprache)
TCP/IP	Transmission Control Protocol/Internet Protocol
TIME	Time Stamp Object (Zeitstempel)
TPDO	Transmit PDO (Sende-PDO)
Widget	Ein Element eines grafischen Benutzerinterfaces (z.B. Bedienknopf, Eingabefeld, ...)

24.

- a -	- t -
Action Registerkarte 11, 38	t_start.tcl 41
	Tcl Kommandos 47
- b -	- u -
Baumstruktur 11	update 44
Bitdarstellung 40	- v -
	Variable global_stop 44
- e -	
edm::addTab 45	
edm::addTestTab 42	
edm::deleteTab 46	
- i -	
Index 11	
- k -	
Kommandos, Tcl 47	
- l -	
Linienreiber 35	
- n -	
NMT Registerkarte 22	
nutzerspezifische Registerkarte 41	
- o -	
Optionen 19	
- p -	
PDO 27	
preop 22	
Preop All 22	
Preop Node 22	
Prozeduren 48	
- r -	
Registerkarte	
Action 11, 38	
NMT 22	
nutzerspezifische 41	
- s -	
Scan Network 22	
Schieberegler 38–39	
Skript laden 48	
start 22	
Start All 22	
Start Node 22	
stop 22	
Subindex 11	
Systemanforderungen 6	

Inhaltsverzeichnis

Einleitung	5
Produktübersicht	5
Hinweise zur Benutzung	6
Unterstützung durch <i>port</i>	6
Schnelleinstieg	8
Erster Programmstart	8
Kommunikation mit einem Gerät	9
Objektverzeichnis-Zugriffe	11
Menüstruktur	14
File	14
Edit	14
View	15
Connection	15
Extras	16
Windows	17
Help	17
Werkzeugleiste	18
Optionen	19
General Settings	19
DCF Settings	20
Network Settings	20
Color Settings	21
Font Settings	21
NMT Kommandos	22
Benutzerdefinierte Skripte	22
Netzwerkübersicht	23
Geräteinfos	23
Description Registerkarte	24
Objektbeschreibung	24
Objektbeschreibungsdatei	24
Overview Registerkarte	26
Index Overview	26
PDO-Konfigurationsregisterkarte	27
Darstellung von PDOs im Chart	27
Process Registerkarte	30
PDO Process Registerkarte	31
File over EtherCAT Registerkarte	32
ESC Viewer Registerkarte	33
EEPROM Viewer Registerkarte	34
Benutzung des Linienschreibers	35
Bedienererweiterungen für den EtherCAT® Device Monitor	38
Schieberegler	38
Schieberegler in einem separatem Fenster	39